```
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG
DDD         DDD  EEEEEEEEEEEE     BBBBBBBBBBBB     UUU         UUU  GGG
DDD         DDD  EEEEEEEEEEEE     BBBBBBBBBBBB     UUU         UUU  GGG
DDD         DDD  EEEEEEEEEEEE     BBBBBBBBBBBB     UUU         UUU  GGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG   GGGGGGGGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG   GGGGGGGGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG   GGGGGGGGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG         GGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG         GGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG         GGG
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUU     GGGGGGGGG
```

```
**FILE**ID**DBGLEVEL1                                  J 16




DDDDDDDD    BBBBBBBB      GGGGGGGG  LL          EEEEEEEEEE  VV      VV  EEEEEEEEEE  LL              11
DDDDDDDD    BBBBBBBB      GGGGGGGG  LL          EEEEEEEEEE  VV      VV  EEEEEEEEEE  LL              11
DD      DD  BB      BB  GG          LL          EE          VV      VV  EE          LL            1111
DD      DD  BB      BB  GG          LL          EE          VV      VV  EE          LL            1111
DD      DD  BB      BB  GG          LL          EE          VV      VV  EE          LL              11
DD      DD  BB      BB  GG          LL          EE          VV      VV  EE          LL              11
DD      DD  BBBBBBBB    GG          LL          EEEEEEEE    VV      VV  EEEEEEEE    LL              11
DD      DD  BBBBBBBB    GG          LL          EEEEEEEE    VV      VV  EEEEEEEE    LL              11
DD      DD  BB      BB  GG  GGGGGG  LL          EE          VV      VV  EE          LL              11
DD      DD  BB      BB  GG  GGGGGG  LL          EE          VV      VV  EE          LL              11
DD      DD  BB      BB  GG      GG  LL          EE            VV  VV    EE          LL              11      ....
DD      DD  BB      BB  GG      GG  LL          EE            VV  VV    EE          LL              11      ....
DDDDDDDD    BBBBBBBB      GGGGGG    LLLLLLLLLL  EEEEEEEEEE      VV      EEEEEEEEEE  LLLLLLLLLL    111111      ....
DDDDDDDD    BBBBBBBB      GGGGGG    LLLLLLLLLL  EEEEEEEEEE      VV      EEEEEEEEEE  LLLLLLLLLL    111111      ....


LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
0 MODULE DBGLEVEL1 (IDENT = 'V04-000') =
1 BEGIN
1 !++
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1 !--
1
1 !++
1 ! FACILITY:     DEBUG (DBG)
1 !
1 ! ABSTRACT:
1 !       This module contains all the miscellaneous routines left over from
1 !       the early days of the debugger. That is, the debugger was mostly
1 !       rewritten in 1982-1983 but after the rewrite there were still
1 !       a handful of routines in different modules that were still used.
1 !       These routines have all been lumped together in this one module.
1 !
1 ! Version:      4.0
1 !
1 ! History:
1 !       Created by:
1 !               R. Title , May 1983
1 !
1
1 ! Require files:
1 !
1 REQUIRE 'SRC$:DBGPROLOG.REQ';
1 LIBRARY 'LIB$:DBGGEN.L32';
1
1 ! Table of contents:
1 !
1 FORWARD ROUTINE
1       dbg$end_of_cmd : NOVALUE,       ! END OF COMMAND PROCESSING ROUTINE
1       dbg$end_of_line : NOVALUE,      ! end of line processing routine
1       dbg$write_mem,                  ! WRITES data TO MEMORY
1       dbg$set_context: NOVALUE,       ! ROUTINE TO INITIALIZE CONTEXT BITS
1       dbg$init_debug : NOVALUE,       ! ROUTINE TO INITIALIZE DEBUG UNDER STARLET
```

```
;    58     0191  1          dbg$cis_connecticf : NOVALUE,    ! Places icf into input stream
;    59     0192  1          dbg$cis_remove : NOVALUE,        ! Remove a link from the cis
;    60     0193  1          dbg$cis_add : NOVALUE,           ! Add a link to the cis
;    61     0194  1          dbg$ins_opcodes: NOVALUE,
;    62     0195  1          dbg$conv_r_50,                   ! SPECIAL-PURPOSE RAD50 CONVERSION ROUTINE.
;    63     0196  1          dbg$out_regname,                 ! Match and symbolize register names
;    64     0197  1          dbg$reg_match,                   ! MATCHES A STRING TO A REGISTER NAME
;    65     0198  1          dbg$digit_scan,                  ! scan string for valid numeric
;    66     0199  1          dbg$output_psl      : NOVALUE,   ! ROUTINE TO OUTPUT PSL IN SPECIAL FORMAT
;    67     0200  1          dbg$map_to_reg_addr,             ! Tries to map input address to an address
;    68     0201  1                                           ! in the reg area in user_runframe
;    69     0202  1          dbg$exact_map_to_reg,            ! Tries to map input address to the address
;    70     0203  1                                           ! of a reg in user_runframe
;    71     0204  1          DBG$LANGUAGE,                    ! Produce name of given language.
;    72     0205  1          DBG$SET_LANG;                    ! Changes the DEBUG syntax
;    73     0206  1
;    74     0207  1  ! Externals
;    75     0208  1  !
;    76     0209  1  EXTERNAL ROUTINE
;    77     0210  1          dbg$ins_decode,                              ! ROUTINE TO OUTPUT MEMORY AS
;    78     0211  1                                                       !  SYMBOLIC INSTRUCTIONS.
;    79     0212  1          dbg$ins_encode,                              ! routine to encode a symbolic instruction
;    80     0213  1          dbg$newline : NOVALUE,                       ! ACTUALLY DO TERMINAL I/O.
;    81     0214  1          dbg$pop_tempmem: NOVALUE,                    ! Pop a temporary memory pool
;    82     0215  1          dbg$push_tempmem,                            ! Create a new temporary memory pool
;    83     0216  1          dbg$sta_getsourcemod,                        ! looks up module rst pointer
;    84     0217  1                                                       ! given the RST pointer
;    85     0218  1          dbg$src_type_lnum_source : NOVALUE,          ! types a range of source line nums
;    86     0219  1          dbg$src_type_pc_source : NOVALUE,            ! types source for a range
;    87     0220  1                                                       ! of PCs.
;    88     0221  1          dbg$def_pr_entry,                            ! Procedure entry code
;    89     0222  1          dbg$get_tempmem,                             ! allocate temporary memory
;    90     0223  1          dbg$ncis_add,                                ! add a CIS to the cis chain
;    91     0224  1
;    92     0225  1          DBG$CANCEL_LOC_VAL: NOVALUE,    ! Cancels '.' and '\'
;    93     0226  1          dbg$check_prot,                              ! CHECKS THE PROTECTION OF A PAGE
;    94     0227  1          DBG$EVENT_INITIALIZATION : NOVALUE,          ! Initialize event structures
;    95     0228  1          dbg$fao_out: NOVALUE,
;    96     0229  1          DBG$FLUSHBUF: NOVALUE,                       ! Flush the print buffer
;    97     0230  1          dbg$get_memory,                              ! Allocate permanent memory
;    98     0231  1          dbg$init_define: NOVALUE,                    ! Initializes define settings
;    99     0232  1          dbg$init_memory: NOVALUE,                    ! Initialize the free memory pool
;   100     0233  1          dbg$init_modes,                              ! INITIALIZES MODES
;   101     0234  1          dbg$init_search: NOVALUE,                    ! Initializes search settings
;   102     0235  1          DBG$NCANCEL_LOC_AND_VAL : NOVALUE,           ! Cancels '.' and '\'
;   103     0236  1          DBG$NCHANGE_TO_NEW : NOVALUE,   ! Switch to new debugger
;   104     0237  1          dbg$ncis_remove,
;   105     0238  1          dbg$nget_trans_radix,                        ! Translate radix
;   106     0239  1          DBG$PARSER_SET_LANGUAGE : NOVALUE,           ! Set up parse table for new language
;   107     0240  1          dbg$print : NOVALUE,                         ! FORMATTED BUFFERED OUTPUT.
;   108     0241  1          DBG$PRINT_CONTROL,                           ! Set print control functions
;   109     0242  1          dbg$read_access,                             ! verify access to memory
;   110     0243  1          dbg$redo_prot,                               ! RESETS THE PROTECTION OF A PAGE TO READ ONLY
;   111     0244  1          dbg$rel_memory : NOVALUE,                    ! Release memory
;   112     0245  1          DBG$REL_TEMPMEM: NOVALUE,                    ! Release all temporary memory
;   113     0246  1          DBG$RST_TEMP_RELEASE: NOVALUE,               ! Release temporary RST entries
;   114     0247  1          dbg$set_define_def: NOVALUE,                 ! Initializes DEFINE data struc
;  115        0248  1              dbg$set_define_lvl: NOVALUE,          ! Sets define level back
;  116        0249  1              DBG$SET_MOD_DEF,                      ! initializes mode settings
;  117        0250  1              dbg$set_mod_lvl,                      ! SETS MODE pointer
;  118        0251  1              dbg$set_out_def : NOVALUE,            ! Initializes OUTPUT config.
;  119        0252  1              DBG$SET_SEARCH_DEF : NOVALUE,   ! inititialize search settings
;  120        0253  1              dbg$set_search_lvl: NOVALUE,          ! Sets search level back
;  121        0254  1              DBG$SET_STP_DEF,                      ! initializes step settings
;  122        0255  1              dbg$sta_setcontext : NOVALUE,         ! Sets registers context
;  123        0256  1              dbg$sta_symname : NOVALUE,            ! Get symbol's name
;  124        0257  1              for$cnv_in_defg,                      ! converts a floating or real
;  125        0258  1              smg$create_key_table,                 ! /Initialize data structures
;  126        0259  1              smg$create_virtual_keyboard,          ! \   used for keypad input.
;  127        0260  1              sys$trnlog,                           ! Translate logical name
;  128        0261  1              dbg$sta_symvalue : NOVALUE,
;  129        0262  1              dbg$free_mem_left,                    ! Longwords remaining in free storage.
;  130        0263  1              dbg$nmake_arg_vect,
;  131        0264  1              dbg$nout_info,
;  132        0265  1              for$cnv_out_i,                        ! Converts integer to ascii string.
;  133        0266  1              dbg$npathdesc_to_cs     : NOVALUE,    ! Get full name of data item
;  134        0267  1              lib$get_ef,                           ! Get event flag
;  135        0268  1              lib$free_ef;                          ! Free event flag
;  136        0269  1
;  137        0270  1 EXTERNAL
;  138        0271  1              dbg$gb_set_break_flag: BYTE,          ! Flag set to true when parsing
;  139        0272  1                                                    !    a SET BREAK command.
;  140        0273  1              dbg$gb_radix: VECTOR[3,BYTE],         ! Radix settings
;  141        0274  1              dbg$gl_context: BITVECTOR,            ! CONTEXT WORD
;  142        0275  1              dbg$gl_developer: BITVECTOR,          ! Developer flags
;  143        0276  1              dbg$gl_ind_com_file: REF VECTOR[,BYTE ! Points to counted string with
;  144        0277  1                                                    !    indirect command file name
;  145        0278  1              dbg$gl_inpfab: BLOCK [, BYTE],        ! FAB FOR 'INPUT'
;  146        0279  1              dbg$gl_inprab: BLOCK [, BYTE],        ! RAB FOR 'INPUT'
;  147        0280  1              dbg$gl_outpfab: BLOCK [, BYTE],       ! FAB FOR 'OUTPUT'
;  148        0281  1              dbg$gl_outprab: BLOCK [, BYTE],       ! RAB FOR 'OUTPUT'
;  149        0282  1              dbg$gl_symhead,                       ! LIST HEAD FOR SYMBOL TABLE
;  150        0283  1              dbg$gl_global_define_ptr,             ! Head of DEFINE list for
;  151        0284  1                                                    !    globally defined symbols
;  152        0285  1              dbg$gl_local_define_ptr,              ! Head of DEFINE list for
;  153        0286  1                                                    !    locally defined symbols
;  154        0287  1              dbg$gl_lis_ptr,
;  155        0288  1              dbg$gl_key_table_id,                  ! Used for DEFINE/KEY
;  156        0289  1              dbg$gl_keyboard_id,                   ! Used for DEFINE/KEY
;  157        0290  1              dbg$gb_keypad_input: BYTE,            ! TRUE if we are trying to do
;  158        0291  1                                                    !    keypad input.
;  159        0292  1              dbg$src_term_width,                   ! Terminal set width
;  160        0293  1              dbg$gb_exc_bre_flag: BYTE,            ! TRUE during an exception break
;  161        0294  1              dbg$gb_go_arg_flag: BYTE,             ! TRUE if there is an argument
;  162        0295  1                                                    !    to GO.
;  163        0296  1              dbg$gl_help_input,                    ! Pointer to HELP input
;  164        0297  1              dbg$gb_search_ptr: REF VECTOR[,BYTE], ! Pointer to search structure
;  165        0298  1              dbg$gb_mod_ptr: REF VECTOR [,BYTE],   ! Pointer to the mode structure
;  166        0299  1              dbg$gb_set_module_flag: BYTE,         ! TRUE during SET MODULE command.
;  167        0300  1              dbg$gb_resignal: BYTE,                ! FLAG FOR RESIGNALING EXCEPTIONS
;  168        0301  1              dbg$gb_take_cmd: BYTE,                ! FLAG WHICH SAYS CONTINUE TO ACCEPT COMMANDS
;  169        0302  1              dbg$gw_loclngth: word,                ! Length field of command override type
;  170        0303  1              dbg$gl_dimenlst : VECTOR,             ! FORTRAN dimension list
;  171        0304  1              dbg$gl_nest_level,                    ! Nesting level of subscripts
```

```
1           dbg$gl_nest_stack: VECTOR,               ! Stack of saved subscripts
1           dbg$gl_search_verb,                      ! Head of command execution tree
1                                                    !    for SEARCH
1           dbg$gl_set_source,
1           dbg$gl_set_source2,
1           dbg$gl_current_primary,                  ! The primary being processed
1           dbg$gl_list: VECTOR,                     ! LIST FOR EXPRESSIONS
1           dbg$gl_loctyp,                           ! command override type.
1           dbg$gl_dflttyp,                          ! type given in SET TYPE.
1           dbg$gl_gbltyp,                           ! type given in SET TYPE/OVERRIDE.
1           dbg$gl_stk : semantic_stack,             ! semantic stack for tokens etc.
1           DBG$GB_STP_PTR :     REF EVENT$STEPPING_DESCRIPTOR, ! POINTER TO CURRENT STEP TYPE
1           dbg$gl_step_num,                         ! number of steps to take in single step mode
1           dbg$pseudo_prog,                         ! Used for DEBUG's CALL command.
1           dbg$gl_logfab : BLOCK [,BYTE],           ! FAB for LOG file
1           dbg$gl_cishead : REF cis$link,           ! Head of command input stream
1           dbg$gl_modrstptr2,                       ! Holds module pointer during
1                                                    ! TYPE command.
1           dbg$gl_module,                           ! Hold module pointer during
1                                                    ! SET SOURCE/MODULE= command.
1           dbg$gl_dirlist,                          ! Holds pointer to directory
1                                                    ! list during
1                                                    ! SET SOURCE dir-list
1                                                    ! command.
1           dbg$src_left_margin,                     ! left margin for source display
1           dbg$src_right_margin,                    ! right margin for source display
1           dbg$gv_control   : dbg$control_flags,    ! DEBUG control bits
1           dbg$gw_gbllngth : WORD,                  ! OVERRIDE LENGTH
1           dbg$gl_next_loc,                         ! NEXT location TO DISPLAY
1           dbg$gb_language : BYTE,                  ! LANGUAGE INDEX
1           dbg$reg_values : VECTOR,                 ! Context regs save area
1           dbg$runframe : BLOCK [, BYTE],           ! current run frame
1           dbg$src_next_modrstptr,                  ! module pointer used by
1                                                    ! dbg$type_cmd.
1           dbg$src_next_lnum,                       ! Contains next line num to
1                                                    ! typed if no line num is
1                                                    ! specified in the TYPE
1                                                    ! command.
1           dbg$src_next_stmt,                       ! as above with stmt num
1           DBG$GL_GET_LEX,                          ! Holds name of current get lex routine
1           DBG$GL_PARTBPTR : VECTOR,                ! List of parse table addresses
1           DBG$GL_REDUC_RT,                         ! Name of action routine for a syntax
1           dbg$gb_def_out : VECTOR [,BYTE],         ! Current OUTPUT configuration
1           dbg$gw_dflteng : WORD,                   ! The length specified in a SET TYPE statement.
1           rst$start_addr: REF rst$entry,           ! Pointer to the module chain (MC).
1           dbg$gl_asci_len,                         ! Length of ascii string.
1           dbg$gb_loc_type: BYTE,                   ! TYPE OF LAST LOCATION EXAMINED
1           dbg$gl_csp_ptr,                          ! pointer to current scope
1           dbg$gl_last_loc,                         ! CURRENT LOCATION
1           dbg$gl_last_val;                         ! CURRENT VALUE
1
1 ! Link symbol saying whether we are linking a debugger to run on a
1 ! version 3B system.
1 !
1 EXTERNAL LITERAL
1           dbg$gl_3b_system: WEAK;
```

```
1 GLOBAL ROUTINE DBG$NCOB_PATHDESC_TO_CS(pathname,name_string) : NOVALUE =
2     BEGIN
2     MAP pathname          : REF pth$pathname;
2
2     LOCAL
2         name_vector       : REF VECTOR[,LONG],
2         name_count,
2         top_name          : REF VECTOR[,BYTE],
2         sub_name          : REF VECTOR[,BYTE],
2         pointer,length;
2
2     name_vector = pathname[pth$a_pathvector];
2     name_vector = name_vector[.pathname[pth$b_pathcnt]];
2     name_count  = .pathname[pth$b_totcnt] - .pathname[pth$b_pathcnt];
2     pathname[pth$b_totcnt]   = .pathname[pth$b_pathcnt];
2     dbg$npathdesc_to_cs(.pathname,top_name);
2     length = .(.top_name)<0,8,0>;
2     DECR index FROM .name_count-1 TO 0 DO
3         BEGIN
3         sub_name = .name_vector[.index];
3         IF .(.sub_name)<0,8,0> GTR 0 THEN length=.length+.(.sub_name)<0,8,0>+4;
2         END;
2     .name_string = pointer = dbg$get_tempmem((.length/%UPVAL)+1);
2     ch$wchar_a(.length,pointer);
2     DECR index FROM .name_count-1 TO 0 DO
3         BEGIN
3         sub_name = .name_vector[.index];
3         IF .(.sub_name)<0,8,0> GTR 0
3         THEN
4             BEGIN
4             ch$move(.(.sub_name)<0,8,0>,sub_name[1],.pointer);
4             pointer = .pointer + .(.sub_name)<0,8,0>;
4             ch$move(4,UPLIT BYTE(' of '),.pointer);
4             pointer = .pointer + 4;
3             END;
2         END;
2     ch$move(.(.top_name)<0,8,0>,top_name[1],.pointer);
1     END;                  ! end of routine dbg$ncob_pathdesc_to_cs


                                          .TITLE  DBGLEVEL1
                                          .IDENT  \V04-000\

                                          .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                 20  66  6F  20  00000 P.AAA:  .ASCII  \ of \                        ;

                                          .EXTRN  DBG$INS_DECODE, DBG$INS_ENCODE
                                          .EXTRN  DBG$NEWLINE, DBG$POP_TEMPMEM
                                          .EXTRN  DBG$PUSH_TEMPMEM
                                          .EXTRN  DBG$STA_GETSOURCEMOD
                                          .EXTRN  DBG$SRC_TYPE_LNUM_SOURCE
                                          .EXTRN  DBG$SRC_TYPE_PC_SOURCE
                                          .EXTRN  DBG$DEF_PR_ENTRY
                                          .EXTRN  DBG$GET_TEMPMEM
                                          .EXTRN  DBG$NCIS_ADD, DBG$CANCEL_LOC_VAL
                                          .EXTRN  DBG$CHECK_PROT, DBG$EVENT_INITIALIZATION
```

```
        .EXTRN  DBG$FAO_OUT, DBG$FLUSHBUF
        .EXTRN  DBG$GET_MEMORY, DBG$INIT_DEFINE
        .EXTRN  DBG$INIT_MEMORY
        .EXTRN  DBG$INIT_MODES, DBG$INIT_SEARCH
        .EXTRN  DBG$NCANCEL_LOC_AND_VAL
        .EXTRN  DBG$NCHANGE_TO_NEW
        .EXTRN  DBG$NCIS_REMOVE
        .EXTRN  DBG$NGET_TRANS_RADIX
        .EXTRN  DBG$PARSER_SET_LANGUAGE
        .EXTRN  DBG$PRINT, DBG$PRINT_CONTROL
        .EXTRN  DBG$READ_ACCESS
        .EXTRN  DBG$REDO_PROT, DBG$REL_MEMORY
        .EXTRN  DBG$REL_TEMPMEM
        .EXTRN  DBG$RST_TEMP_RELEASE
        .EXTRN  DBG$SET_DEFINE_DEF
        .EXTRN  DBG$SET_DEFINE_LVL
        .EXTRN  DBG$SET_MOD_DEF
        .EXTRN  DBG$SET_MOD_LVL
        .EXTRN  DBG$SET_OUT_DEF
        .EXTRN  DBG$SET_SEARCH_DEF
        .EXTRN  DBG$SET_SEARCH_LVL
        .EXTRN  DBG$SET_STP_DEF
        .EXTRN  DBG$STA_SETCONTEXT
        .EXTRN  DBG$STA_SYMNAME
        .EXTRN  FOR$CNV_IN_DEFG
        .EXTRN  SMG$CREATE_KEY_TABLE
        .EXTRN  SMG$CREATE_VIRTUAL_KEYBOARD
        .EXTRN  SYS$TRNLOG, DBG$STA_SYMVALUE
        .EXTRN  DBG$FREE_MEM_LEFT
        .EXTRN  DBG$NMAKE_ARG_VECT
        .EXTRN  DBG$NOUT_INFO, FOR$CNV_OUT_I
        .EXTRN  DBG$NPATHDESC_TO_CS
        .EXTRN  LIB$GET_EF, LIB$FREE_EF
        .EXTRN  DBG$GB_SET_BREAK_FLAG
        .EXTRN  DBG$GB_RADIX, DBG$GL_CONTEXT
        .EXTRN  DBG$GL_DEVELOPER
        .EXTRN  DBG$GL_IND_COM_FILE
        .EXTRN  DBG$GL_INPFAB, DBG$GL_INPRAB
        .EXTRN  DBG$GL_OUTPFAB, DBG$GL_OUTPRAB
        .EXTRN  DBG$GL_SYMHEAD, DBG$GL_GLOBAL_DEFINE_PTR
        .EXTRN  DBG$GL_LOCAL_DEFINE_PTR
        .EXTRN  DBG$GL_LIS_PTR, DBG$GL_KEY_TABLE_ID
        .EXTRN  DBG$GL_KEYBOARD_ID
        .EXTRN  DBG$GB_KEYPAD_INPUT
        .EXTRN  DBG$SRC_TERM_WIDTH
        .EXTRN  DBG$GB_EXC_BRE_FLAG
        .EXTRN  DBG$GB_GO_ARG_FLAG
        .EXTRN  DBG$GL_HELP_INPUT
        .EXTRN  DBG$GB_SEARCH_PTR
        .EXTRN  DBG$GB_MOD_PTR, DBG$GB_SET_MODULE_FLAG
        .EXTRN  DBG$GB_RESIGNAL
        .EXTRN  DBG$GB_TAKE_CMD
        .EXTRN  DBG$GW_LOCLNGTH
        .EXTRN  DBG$GL_DIMENLST
        .EXTRN  DBG$GL_NEST_LEVEL
        .EXTRN  DBG$GL_NEST_STACK
        .EXTRN  DBG$GL_SEARCH_VERB
```

```
                                                          .EXTRN  DBG$GL_SET_SOURCE
                                                          .EXTRN  DBG$GL_SET_SOURCE2
                                                          .EXTRN  DBG$GL_CURRENT_PRIMARY
                                                          .EXTRN  DBG$GL_LIST, DBG$GL_LOCTYP
                                                          .EXTRN  DBG$GL_DFLTTYP, DBG$GL_GBLTYP
                                                          .EXTRN  DBG$GL_STK, DBG$GB_STP_PTR
                                                          .EXTRN  DBG$GL_STEP_NUM
                                                          .EXTRN  DBG$PSEUDO_PROG
                                                          .EXTRN  DBG$GL_LOGFAB, DBG$GL_CISHEAD
                                                          .EXTRN  DBG$GL_MODRSTPTR2
                                                          .EXTRN  DBG$GL_MODULE, DBG$GL_DIRLIST
                                                          .EXTRN  DBG$SRC_LEFT_MARGIN
                                                          .EXTRN  DBG$SRC_RIGHT_MARGIN
                                                          .EXTRN  DBG$GV_CONTROL, DBG$GW_GBLLNGTH
                                                          .EXTRN  DBG$GL_NEXT_LOC
                                                          .EXTRN  DBG$GB_LANGUAGE
                                                          .EXTRN  DBG$REG_VALUES, DBG$RUNFRAME
                                                          .EXTRN  DBG$SRC_NEXT_MODRSTPTR
                                                          .EXTRN  DBG$SRC_NEXT_LNUM
                                                          .EXTRN  DBG$SRC_NEXT_STMT
                                                          .EXTRN  DBG$GL_GET_LEX, DBG$GL_PARTBPTR
                                                          .EXTRN  DBG$GL_REDUC_RT
                                                          .EXTRN  DBG$GB_DEF_OUT, DBG$GW_DFLTLENG
                                                          .EXTRN  RST$START_ADDR, DBG$GL_ASCI_LEN
                                                          .EXTRN  DBG$GB_LOC_TYPE
                                                          .EXTRN  DBG$GL_CSP_PTR, DBG$GL_LAST_LOC
                                                          .EXTRN  DBG$GL_LAST_VAL
                                                          .WEAK   DBG$GL_3B_SYSTEM

                                                          .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                07FC 00000                .ENTRY  DBG$NCOB_PATHDESC_TO_CS, Save R2,R3,R4,R5,- ; 0361
                                                                  R6,R7,R8,R9,R10                             ;
               5E           04  C2 00002                  SUBL2   #4, SP                                      ;
               51      04   AC  D0 00005                  MOVL    PATHNAME, R1                                ; 0372
               57      08   A1  9E 00009                  MOVAB   8(R1), NAME_VECTOR                          ;
               50      01   A1  9A 0000D                  MOVZBL  1(R1), R0                                   ; 0373
               57         6740  DE 00011                  MOVAL   (NAME_VECTOR)[R0], NAME_VECTOR              ;
               56           61  9A 00015                  MOVZBL  (R1), NAME_COUNT                            ; 0374
               56           50  C2 00018                  SUBL2   R0, NAME_COUNT                              ;
               61           50  90 0001B                  MOVB    R0, (R1)                                    ; 0375
                    4002    8F  BB 0001E                  PUSHR   #^M<R1,SP>                                  ; 0376
     00000000G 00           02  FB 00022                  CALLS   #2, DBG$NPATHDESC_TO_CS                     ;
               5A           6E  D0 00029                  MOVL    TOP_NAME, R10                               ; 0377
               52           6A  9A 0002C                  MOVZBL  (R10), LENGTH                               ;
               50           56  D0 0002F                  MOVL    NAME_COUNT, INDEX                           ; 0380
                            0E  11 00032                  BRB     2$                                          ;
               59         6740  D0 00034 1$:              MOVL    (NAME_VECTOR)[INDEX], SUB_NAME              ;
               51           69  9A 00038                  MOVZBL  (SUB_NAME), R1                              ; 0381
                            05  15 0003B                  BLEQ    2$                                          ;
               52      04 A142  9E 0003D                  MOVAB   4(R1)[LENGTH], LENGTH                       ;
               EF           50  F4 00042 2$:              SOBGEQ  INDEX, 1$                                   ; 0378
50             52           04  C7 00045                  DIVL3   #4, LENGTH, R0                              ; 0383
                       01   A0  9F 00049                  PUSHAB  1(R0)                                       ;
     00000000G 00           01  FB 0004C                  CALLS   #1, DBG$GET_TEMPMEM                         ;
               58           50  D0 00053                  MOVL    R0, POINTER                                 ;
          08   BC           58  D0 00056                  MOVL    POINTER, @NAME_STRING                       ;
```

```
                        88          52  90 0005A          MOVB    LENGTH, (POINTER)+                            ; 0384
                                    1D  11 0005D          BRB     4$                                            ; 0385
                        59        6746  D0 0005F 3$:      MOVL    (NAME_VECTOR)[INDEX], SUB_NAME                ; 0387
                                    69  95 00063          TSTB    (SUB_NAME)                                    ; 0388
                                    15  13 00065          BEQL    4$                                            ;
                        50          69  9A 00067          MOVZBL  (SUB_NAME), R0                                ; 0391
          68      01    A9          50  28 0006A          MOVC3   R0, 1(SUB_NAME), (POINTER)                    ;
                        50          69  9A 0006F          MOVZBL  (SUB_NAME), R0                                ; 0392
                        58          50  C0 00072          ADDL2   R0, POINTER                                   ;
                        88 00000000'  EF  D0 00075        MOVL    P.AAA, (POINTER)+                             ; 0393
                        E0          56  F4 0007C 4$:      SOBGEQ  INDEX, 3$                                     ; 0385
                        50          6A  9A 0007F          MOVZBL  (R10), R0                                     ; 0397
          68      01    AA          50  28 00082          MOVC3   R0, 1(R10), (POINTER)                         ;
                                        04 00087          RET                                                   ; 0398

; Routine Size:  136 bytes,    Routine Base:  DBG$CODE + 0000
```

```
1 GLOBAL ROUTINE DBG$END_OF_CMD : NOVALUE =
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !       Resets all DEBUG context that is exclusive to a single
1 !       DEBUG command. This includes resetting default
1 !       modes from single line overrides back to the actual default
1 !       modes and resetting a large number of context bits.
1 !
1 !       This routine also releases all temporary memory allocated in the
1 !       course of processing the command, and it releases all unreferenced
1 !       RST entries on the Temporary RST Entry List.
1 !
1 ! FORMAL PARAMETERS:
1 !       none
1 !
1 ! IMPLICIT INPUTS:
1 !       none
1 !
1 ! IMPLICIT OUTPUTS:
1 !       The default modes, step-modes, and context bits are established.
1 !       Some global storage is re-initialized and all excess storage is released.
1 !
1 ! ROUTINE VALUE:
1 !       novalue
1 !
1 ! SIDE EFFECTS:
1 !       none
1 !--
1
2    BEGIN
2
2       !+
2       ! Set the exit flag to true so that if an error occurs during
2       ! the processing of this routine, that error is perceived
2       ! as fatal. This routine guarantees the internal consistency
2       ! of DEBUG, and must succeed or give up.
2       !-
2       dbg$gv_control[dbg$v_control_exit] = TRUE;
2
2       ! Clear the ALLOCATE flag. This is set during SET MODULE/ALLOCATE
2       ! to allow the allocation of additional memory.
2       !
2       dbg$gv_control[dbg$v_control_allocate] = FALSE;
2
2       ! Reset the Print control for DBG$PRINT.  And flush out the print
2       ! buffer.
2       !
2       DBG$PRINT_CONTROL(DBG$K_PRT_RESET);
2       DBG$FLUSHBUF();
2
2       ! Reset mode level to user default level
2       !
2       dbg$init_modes (override_mode, user_def_mode);
2       dbg$set_mod_lvl (user_def_mode);
2
2       ! Reset search settings back to user default level
2       !
```

```
;  325       0456  2         dbg$init_search (override_search, user_def_search);
;  326       0457  2         dbg$set_search_lvl (user_def_search);
;  327       0458  2
;  328       0459  2         ! Reset define settings back to user default level
;  329       0460  2         !
;  330       0461  2         dbg$init_define (override_define, user_def_define);
;  331       0462  2         dbg$set_define_lvl (user_def_define);
;  332       0463  2
;  333       0464  2         dbg$set_context ();
;  334       0465  2
;  335       0466  2         DBG$REL_TEMPMEM();
;  336       0467  2         DBG$RST_TEMP_RELEASE();
;  337       0468  2         dbg$gl_list [0] = 0;              ! Zero out the locations that hold breakpoint setting data.
;  338       0469  2         dbg$gl_list [1] = 0;
;  339       0470  2         dbg$gl_list [2] = 0;
;  340       0471  2         dbg$gl_lis_ptr  = 0;              ! Zero current ptr to command arg list
;  341       0472  2         dbg$gl_asci_len = .dbg$gb_mod_ptr[mode_length]; ! Initialize ascii length
;  342       0473  2         dbg$gl_loctyp = -1;               ! Zero command override type.
;  343       0474  2         dbg$gw_loclngth= 0;               ! And its associated length.
;  344       0475  2
;  345       0476  2         dbg$gl_module = 0;                ! Zero out global used to hold
;  346       0477  2                                           ! module pointer during
;  347       0478  2                                           ! SET SOURCE/MODULE= command.
;  348       0479  2         dbg$gl_modrstptr2 = 0;            ! Zero out global used to hold
;  349       0480  2                                           ! module pointer during
;  350       0481  2                                           ! TYPE command.
;  351       0482  2         dbg$gl_set_source = 0;
;  352       0483  2         dbg$gl_set_source2 = 0;
;  353       0484  2         dbg$gl_current_primary = 0;       ! Clear the current primary cause there isn't one anymore
;  354       0485  2         dbg$gb_set_module_flag = FALSE;   ! This flag is TRUE during a SET MODULE
;  355       0486  2                                           !     command.
;  356       0487  2         zerocor (dbg$gl_dimenlst, 10);    ! Zero storage to hold array dimensions.
;  357       0488  2         zerocor (dbg$gl_nest_stack, 25);  ! Zero storage to hold array dimensions
;  358       0489  2                                           !     during nested subscript evaluation
;  359       0490  2         dbg$gl_nest_level = 0;            ! Nesting level of subscript expressions
;  360       0491  2                                           !     set back to zero.
;  361       0492  2         dbg$gb_set_break_flag = FALSE;    ! Initialize a flag saying whether we
;  362       0493  2                                           ! are in the middle of processing a
;  363       0494  2                                           ! SET BREAK command. This is used in
;  364       0495  2                                           ! DBGPARSER to resolve ambiguities
;  365       0496  2                                           ! involving SET BREAK . DO (command).
;  366       0497  2                                           ! This flag gets set to TRUE in DBGEVENT
;  367       0498  2                                           ! when we discover we are indeed
;  368       0499  2                                           ! processing a SET BREAK command.
;  369       0500  2
;  370       0501  2         dbg$sta_setcontext (0);           ! Set default register context
;  371       0502  2
;  372       0503  2                                           ! Now cancel exit flag since all went well.
;  373       0504  2         dbg$gv_control[dbg$v_control_exit] = FALSE;
;  374       0505  1     END;



                                  007C 00000         .ENTRY  DBG$END_OF_CMD, Save R2,R3,R4,R5,R6          ; 0399
                    56 00000000G  00  9E 00002         MOVAB   DBG$GV_CONTROL, R6                           ;
```

```
                                                                I 1
DBGLEVEL1                                                       16-Sep-1984 01:27:02    VAX-11 Bliss-32 V4.0-742              Page  11
V04-000                                                         14-Sep-1984 12:17:02    [DEBUG.SRC]DBGLEVEL1.B32;1                 (3)

                                     66              10   88 00009          BISB2   #16, DBG$GV_CONTROL                          ; 0436
                                     66        80    8F   8A 0000C          BICB2   #128, DBG$GV_CONTROL                         ; 0441
                                                     05   DD 00010          PUSHL   #5                                           ; 0446
                          00000000G  00              01   FB 00012          CALLS   #1, DBG$PRINT_CONTROL                        ;
                          00000000G  00              00   FB 00019          CALLS   #0, DBG$FLUSHBUF                             ; 0447
                                                     01   DD 00020          PUSHL   #1                                           ; 0451
                                                     02   DD 00022          PUSHL   #2                                           ;
                          00000000G  00              02   FB 00024          CALLS   #2, DBG$INIT_MODES                           ;
                                                     01   DD 0002B          PUSHL   #1                                           ; 0452
                          00000000G  00              01   FB 0002D          CALLS   #1, DBG$SET_MOD_LVL                          ;
                                                     01   DD 00034          PUSHL   #1                                           ; 0456
                                                     02   DD 00036          PUSHL   #2                                           ;
                          00000000G  00              02   FB 00038          CALLS   #2, DBG$INIT_SEARCH                          ;
                                                     01   DD 0003F          PUSHL   #1                                           ; 0457
                          00000000G  00              01   FB 00041          CALLS   #1, DBG$SET_SEARCH_LVL                       ;
                                                     01   DD 00048          PUSHL   #1                                           ; 0461
                                                     02   DD 0004A          PUSHL   #2                                           ;
                          00000000G  00              02   FB 0004C          CALLS   #2, DBG$INIT_DEFINE                          ;
                                                     01   DD 00053          PUSHL   #1                                           ; 0462
                          00000000G  00              01   FB 00055          CALLS   #1, DBG$SET_DEFINE_LVL                       ;
                              0000V  CF              00   FB 0005C          CALLS   #0, DBG$SET_CONTEXT                          ; 0464
                          00000000G  00              00   FB 00061          CALLS   #0, DBG$REL_TEMPMEM                          ; 0466
                          00000000G  00              00   FB 00068          CALLS   #0, DBG$RST_TEMP_RELEASE                     ; 0467
                                     00000000G       00   7C 0006F          CLRQ    DBG$GL_LIST                                  ; 0468
                                     00000000G       00   D4 00075          CLRL    DBG$GL_LIST+8                                ; 0470
                                     00000000G       00   D4 0007B          CLRL    DBG$GL_LIS_PTR                               ; 0471
                                  50 00000000G       00   D0 00081          MOVL    DBG$GB_MOD_PTR, R0                           ; 0472
                          00000000G  00        01    A0   9A 00088          MOVZBL  1(R0), DBG$GL_ASCI_LEN                       ;
                          00000000G  00              01   CE 00090          MNEGL   #1, DBG$GL_LOCTYP                            ; 0473
                                     00000000G       00   B4 00097          CLRW    DBG$GW_LOCLNGTH                              ; 0474
                                     00000000G       00   D4 0009D          CLRL    DBG$GL_MODULE                                ; 0476
                                     00000000G       00   D4 000A3          CLRL    DBG$GL_MODRSTPTR2                            ; 0479
                                     00000000G       00   D4 000A9          CLRL    DBG$GL_SET_SOURCE                            ; 0482
                                     00000000G       00   D4 000AF          CLRL    DBG$GL_SET_SOURCE2                           ; 0483
                                     00000000G       00   D4 000B5          CLRL    DBG$GL_CURRENT_PRIMARY                       ; 0484
                                     00000000G       00   94 000BB          CLRB    DBG$GB_SET_MODULE_FLAG                       ; 0485
          28             00          6E              00   2C 000C1          MOVC5   #0, (SP), #0, #40, DBG$GL_DIMENLST           ; 0487
                                     00000000G       00      000C6                                                               ;
   0064   8F             00          6E              00   2C 000CB          MOVC5   #0, (SP), #0, #100, DBG$GL_NEST_STACK        ; 0488
                                     00000000G       00      000D2                                                               ;
                                     00000000G       00   D4 000D7          CLRL    DBG$GL_NEST_LEVEL                            ; 0490
                                     00000000G       00   94 000DD          CLRB    DBG$GB_SET_BREAK_FLAG                        ; 0492
                                                     7E   D4 000E3          CLRL    -(SP)                                        ; 0501
                          00000000G  00              01   FB 000E5          CALLS   #1, DBG$STA_SETCONTEXT                       ;
                                     66              10   8A 000EC          BICB2   #16, DBG$GV_CONTROL                          ; 0504
                                                          04 000EF          RET                                                  ; 0505

; Routine Size:  240 bytes,    Routine Base:  DBG$CODE + 0088
;   376          0506  1 GLOBAL ROUTINE dbg$end_of_line : NOVALUE =
;   377          0507  1 !++
;   378          0508  1 ! FUNCTIONAL DESCRIPTION:
;   379          0509  1 !       Calls dbg$end_of_cmd to reset all single command context.
;   380          0510  1 !       Then frees the storage that was allocated to hold the command
;   381          0511  1 !       line. The top link of the command input stream is removed, but
;   382          0512  1 !       only if it is of type "buffer".
;   383          0513  1 !
;   384          0514  1 ! FORMAL PARAMETERS:
;   385          0515  1 !       none
;   386          0516  1 !
;   387          0517  1 ! IMPLICIT INPUTS:
;   388          0518  1 !       The head of the command argument list.
;   389          0519  1 !
;   390          0520  1 ! IMPLICIT OUTPUTS:
;   391          0521  1 !       none
;   392          0522  1 !
;   393          0523  1 ! ROUTINE VALUE:
;   394          0524  1 !       none
;   395          0525  1 !
;   396          0526  1 ! SIDE EFFECTS:
;   397          0527  1 !       Defaults are reestablished. Storage for input line is freed.
;   398          0528  1 !       A link may be removed from the command argument list.
;   399          0529  1 !--
;   400          0530  2     BEGIN
;   401          0531  2     LOCAL
;   402          0532  2         type;
;   403          0533  2
;   404          0534  2         dbg$end_of_cmd ();                  ! Perform end of command cleanup
;   405          0535  2         dbg$gv_control[dbg$v_control_exit] = TRUE;      ! Set the exit flag to return to CLI on errors
;   406          0536  2
;   407          0537  2         ! We only want to remove the top link of the cis if that link is a
;   408          0538  2         ! buffer of some flavor. If the top link is of type cis_rab, it has
;   409          0539  2         ! just been put there by an @... command and not yet read from.
;   410          0540  2
;   411          0541  2         type = .dbg$gl_cishead [cis$b_input_type];
;   412          0542  2         IF .type EQL cis_inpbuf
;   413          0543  2         OR .type EQL cis_acbuf
;   414          0544  2         OR .type EQL cis_while
;   415          0545  2         OR .type EQL cis_repeat
;   416          0546  2         OR .type EQL cis_if
;   417          0547  2         THEN
;   418          0548  2                 dbg$cis_remove();
;   419          0549  2
;   420          0550  2         dbg$gv_control[dbg$v_control_exit] = FALSE;     ! Reset exit flag.
;   421          0551  2
;   422          0552  1     END;



                                      0004 00000        .ENTRY  DBG$END_OF_LINE, Save R2                        ; 0506
                   52 00000000G   00   9E 00002        MOVAB   DBG$GV_CONTROL, R2                              ;
              FF02 CF             00   FB 00009        CALLS   #0, DBG$END_OF_CMD                              ; 0534
                   62             10   88 0000E        BISB2   #16, DBG$GV_CONTROL                             ; 0535
                   50 00000000G   00   D0 00011        MOVL    DBG$GL_CISHEAD, R0                              ; 0541
```

```
                    50        02    A0  9A 00018          MOVZBL  2(R0), TYPE                                  ;
                    02              50  D1 0001C          CMPL    TYPE, #2                                     ; 0542
                                    14  13 0001F          BEQL    1$                                           ;
                    03              50  D1 00021          CMPL    TYPE, #3                                     ; 0543
                                    0F  13 00024          BEQL    1$                                           ;
                    05              50  D1 00026          CMPL    TYPE, #5                                     ; 0544
                                    0A  13 00029          BEQL    1$                                           ;
                    04              50  D1 0002B          CMPL    TYPE, #4                                     ; 0545
                                    05  13 0002E          BEQL    1$                                           ;
                    06              50  D1 00030          CMPL    TYPE, #6                                     ; 0546
                                    05  12 00033          BNEQ    2$                                           ;
              0000V CF              00  FB 00035 1$:      CALLS   #0, DBG$CIS_REMOVE                           ; 0548
                    62              10  8A 0003A 2$:      BICB2   #16, DBG$GV_CONTROL                          ; 0550
                                        04 0003D          RET                                                  ; 0552

; Routine Size:  62 bytes,    Routine Base:  DBG$CODE + 0178


;  423          0553  1
```

```
GLOBAL ROUTINE dbg$write_mem (dest_address, src_address, length) =
!++
! FUNCTIONAL DESCRIPTION:
!       Writes a sequence of values (bytes) to memory in
!       the user program.  The destination, source, and
!       number of bytes to write are all passed as parameters.
!
!       THE PROTECTION OF THE FIRST BYTE TO BE WRITTEN AND THE LAST
!       BYTE TO BE WRITTEN ARE BOTH CHECKED. THE STATUS OF BOTH PAGES
!       (THEY MAY BE THE SAME PAGE) IS SAVED. THEN THE VALUE IS WRITTEN
!       TO THE ADDRESS (THE PAGE PROTECTION IS CHANGED DURING THE
!       CHECKING OPERATION).
!
!       THEN, IF THE PROTECTION WAS CHANGED IN EITHER CASE, THE
!       PROTECTION IS REESTABLISHED. IF EVERYTHING WAS SUCCESSFUL,
!       THE ROUTINE RETURNS TRUE. OTHERWISE, IT RETURNS FALSE.
!
! Formal Parameters:
!       dest_address    - THE ADDRESS OF THE LOCATION TO BE CHANGED
!       src_address     - The address of where the bytes are stored.
!       length          - The number of bytes to be written.
!
! IMPLICIT INPUTS:
!       None.
!
! IMPLICIT OUTPUTS:
!       THE PAGE PROTECTION MAY BE MOMENTARILY ALTERED, THEN REINSTALLED.
!
! ROUTINE VALUE:
!       TRUE OR FALSE
!
! SIDE EFFECTS:
!       THE VALUE IS WRITTEN TO MEMORY
!--
        BEGIN

        MAP
                dest_address    : REF VECTOR[,BYTE],
                src_address     : REF VECTOR[,BYTE];

        LOCAL
                prot_status_1,
                protection_1: BYTE,
                prot_status_2,
                protection_2: BYTE;

        IF ((prot_status_1 = dbg$check_prot (.dest_address, protection_1)) NEQ 0)
                AND ((prot_status_2 = dbg$check_prot (.dest_address + .length - 1, protection_2)) NEQ 0)
        THEN
                BEGIN

                !++
                ! PROTECTION HAS EITHER BEEN ALTERED SUCCESSFULLY, OR IT
                ! DID NOT NEED TO BE ALTERED. NOW WRITE THE VALUE INTO THE
                ! ADDRESS.
                !--

```

```
;  482         0611  3              CH$MOVE (.length, src_address [0], dest_address [0]);
;  483         0612  3
;  484         0613  3              !++
;  485         0614  3              ! IF EITHER OF THE PROTECTION STATUSES SAY RESET THE PROTECTION
;  486         0615  3              ! (TO READ ONLY), THEN RESET THE PROTECTION ON THAT PAGE.
;  487         0616  3              !--
;  488         0617  3
;  489         0618  3              IF .prot_status_1 EQL dbg$k_reset_prt
;  490         0619  3              THEN
;  491         0620  4                  BEGIN
;  492         0621  4                  dbg$redo_prot (.dest_address, protection_1);
;  493         0622  3                  END;
;  494         0623  3
;  495         0624  3              IF .prot_status_2 EQL dbg$k_reset_prt
;  496         0625  3              THEN
;  497         0626  4                  BEGIN
;  498         0627  4                  dbg$redo_prot (.dest_address + .length - 1, protection_2);
;  499         0628  3                  END;
;  500         0629  3
;  501         0630  3              RETURN TRUE                          ! User program updated correctly
;  502         0631  3              END
;  503         0632  2          ELSE
;  504         0633  2              RETURN FALSE
;  505         0634  2
;  506         0635  1          END;



                                         07FC 00000          .ENTRY  DBG$WRITE_MEM, Save R2,R3,R4,R5,R6,R7,R8,-   ; 0554
                                                                     R9,R10                                        ;
                  5A 00000000G   00   9E 00002          MOVAB   DBG$CHECK_PROT, R10                               ;
                  59 00000000G   00   9E 00009          MOVAB   DBG$REDO_PROT, R9                                 ;
                  5E             08   C2 00010          SUBL2   #8, SP                                            ;
                                 5E   DD 00013          PUSHL   SP                                                ; 0600
                  56        04   AC   D0 00015          MOVL    DEST_ADDRESS, R6                                  ;
                                 56   DD 00019          PUSHL   R6                                                ;
                  6A             02   FB 0001B          CALLS   #2, DBG$CHECK_PROT                                ;
                  58             50   D0 0001E          MOVL    R0, PROT_STATUS_1                                 ;
                                 3C   13 00021          BEQL    3$                                                ;
                            04   AE   9F 00023          PUSHAB  PROTECTION_2                                      ; 0601
      50          56        0C   AC   C1 00026          ADDL3   LENGTH, R6, R0                                    ;
                            FF   A0   9F 0002B          PUSHAB  -1(R0)                                            ;
                  6A             02   FB 0002E          CALLS   #2, DBG$CHECK_PROT                                ;
                  57             50   D0 00031          MOVL    R0, PROT_STATUS_2                                 ;
                                 29   13 00034          BEQL    3$                                                ;
      66     08   BC        0C   AC   28 00036          MOVC3   LENGTH, @SRC_ADDRESS, (R6)                        ; 0611
                  02             58   D1 0003C          CMPL    PROT_STATUS_1, #2                                 ; 0618
                                 07   12 0003F          BNEQ    1$                                                ;
                          4040   8F   BB 00041          PUSHR   #^M<R6,SP>                                        ; 0621
                  69             02   FB 00045          CALLS   #2, DBG$REDO_PROT                                 ;
                  02             57   D1 00048  1$:     CMPL    PROT_STATUS_2, #2                                 ; 0624
                                 0E   12 0004B          BNEQ    2$                                                ;
                            04   AE   9F 0004D          PUSHAB  PROTECTION_2                                      ; 0627
      50          56        0C   AC   C1 00050          ADDL3   LENGTH, R6, R0                                    ;
                            FF   A0   9F 00055          PUSHAB  -1(R0)                                            ;
```

```
                                        69            02  FB 00058          CALLS   #2, DBG$REDO_PROT                          ;
                                        50            01  D0 0005B 2$:      MOVL    #1, R0                                     ; 0633
                                                          04 0005E          RET                                                ;
                                                      50  D4 0005F 3$:      CLRL    R0                                         ;
                                                          04 00061          RET                                                ; 0635

; Routine Size:  98 bytes,    Routine Base:  DBG$CODE + 01B6
```

```
1 GLOBAL ROUTINE dbg$set_context : NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !       initializes context bits that are necessary for command
1 !       processing. These bits are valid only during the processing
1 !       of a single command. They are all reset after each command.
1 !
1 ! CALLING SEQUENCE:
1 !       dbg$set_context ()
1 !
1 ! INPUTS:
1 !       none
1 !
1 ! IMPLICIT INPUTS:
1 !       the names of the context bits that are to be turned off
1 !
1 ! OUTPUTS:
1 !       none
1 !
1 ! IMPLICIT OUTPUTS:
1 !       none
1 !
1 ! ROUTINE VALUE:
1 !       novalue
1 !
1 ! SIDE EFFECTS:
1 !       the context bits are set to false
1 !--
1
2       BEGIN
2       dbg$gl_context [dbg$k_all]       = FALSE;
2       dbg$gl_context [dbg$k_all_break] = FALSE;
2       dbg$gl_context [dbg$k_all_trace] = FALSE;
2       dbg$gl_context [dbg$k_all_watch] = FALSE;
2       dbg$gl_context [dbg$k_break]     = FALSE;
2       dbg$gl_context [dbg$k_cancel]    = FALSE;
2       dbg$gl_context [dbg$k_examine]   = FALSE;
2       dbg$gl_context [dbg$k_language]  = FALSE;
2       dbg$gl_context [dbg$k_mode]      = FALSE;
2       dbg$gl_context [dbg$k_module]    = FALSE;
2       dbg$gl_context [dbg$k_override]  = FALSE;
2       dbg$gl_context [dbg$k_resignal]  = FALSE;
2       dbg$gl_context [dbg$k_scope]     = FALSE;
2       dbg$gl_context [dbg$k_search]    = FALSE;
2       dbg$gl_context [dbg$k_set_break] = FALSE;
2       dbg$gl_context [dbg$k_step]      = FALSE;
2       dbg$gl_context [dbg$k_trce_call] = FALSE;
2       dbg$gl_context [dbg$k_trace]     = FALSE;
2       dbg$gl_context [dbg$k_traceback] = FALSE;
2       dbg$gl_context [dbg$k_watch]     = FALSE;
2       dbg$gl_context [dbg$k_trce_brch] = FALSE;
2       dbg$gl_context [dbg$k_thread]    = FALSE;
2       dbg$gl_context [dbg$k_output]    = FALSE;
2       dbg$gl_context [dbg$k_log]       = FALSE;
2       dbg$gl_context [dbg$k_source]    = FALSE;
2       dbg$gl_context [dbg$k_margins]   = FALSE;
```

```
;  565          0693  2        dbg$gl_context [dbg$k_maxfiles]  = FALSE;
;  566          0694  1        END;



                                             0000 00000        .ENTRY  DBG$SET_CONTEXT, Save nothing                   ; 0636
000000000G  00          1B          00          00  F0 00002        INSV    #0, #0, #27, DBG$GL_CONTEXT                   ; 0693
                                               04 0000B        RET                                                    ; 0694

; Routine Size:  12 bytes,    Routine Base:  DBG$CODE + 0218
;  1  GLOBAL ROUTINE DBG$INIT_DEBUG: NOVALUE =
;  1  !
;  1  ! FUNCTION
;  1  !       This routine drives the DEBUG initialization when DEBUG first comes up.
;  1  !
;  1  ! INPUTS
;  1  !       NONE
;  1  !
;  1  ! OUTPUTS
;  1  !       NONE
;  1  !
;  1
;  2     BEGIN
;  2
;  2     BIND
;  2         DBG_INPUT_DEVICE        = UPLIT BYTE (%ASCII 'DBG$INPUT'),
;  2         DBG_INP_DEV_SIZE        = %CHARCOUNT (%ASCII 'DBG$INPUT'),
;  2         DBG_OUTPUT_DEVICE       = UPLIT BYTE (%ASCII 'DBG$OUTPUT'),
;  2         DBG_OUT_DEV_SIZE        = %CHARCOUNT (%ASCII 'DBG$OUTPUT'),
;  2
;  2         SYS_INPUT_DEVICE        = UPLIT BYTE (%ASCII 'SYS$INPUT'),
;  2         SYS_INP_DEV_SIZE        = %CHARCOUNT (%ASCII 'SYS$INPUT'),
;  2         SYS_OUTPUT_DEVICE       = UPLIT BYTE (%ASCII 'SYS$OUTPUT'),
;  2         SYS_OUT_DEV_SIZE        = %CHARCOUNT (%ASCII 'SYS$OUTPUT');
;  2
;  2     LOCAL
;  2         DEF_RADIX,                          ! Default radix
;  2         DEVCHAR: REF BLOCK[,BYTE],          ! Device characteristics field
;  2         DUMMY: VECTOR[2],                   ! Output area for $TRNLOG
;  2         DUMMY_BUFFER: VECTOR[256,BYTE],     ! ''
;  2         EVNT_FLAG,
;  2         FILESPEC: DBG$STG_DESC,             ! String descriptor
;  2         HEADER: REF DEFINE$HEADER,          ! Header block for define
;  2                                             !     symbol table.
;  2         ITEM: BLOCK[6, LONG],               ! Item list for $GETSYI
;  2         LEN,
;  2         OPEN_STATUS,                        ! Save the failing status from $OPEN
;  2                                             !     DBG$INPUT
;  2         OUTPUT_STATUS,                      ! Save the failing status from $CREATE
;  2                                             !     DBG$OUTPUT
;  2         SDBGINIT_STGDESC: BLOCK[8,BYTE],    ! String descriptor
;  2         SDBGINIT_STG: VECTOR [9, BYTE],     ! String in string descriptor
;  2         STATUS,
;  2         STATUS1,
;  2         VERSION_BUFFER: VECTOR[8, BYTE];
;  2
;  2
;  2
;  2     ! Initialize an area of free storage.  This must be done first since many
;  2     ! of the things below will call the memory allocation routines.
;  2     !
;  2     DBG$INIT_MEMORY();
;  2
;  2
;  2     ! Initialize the bit that says whether we are on a V4 system.
;  2     ! We call the system service $GETSYI to find out this information.
;  2     !
```

```
! Note - the code to call GETSYI is commented out because this
! turned out to be unreliable (could get back a variety of things,
! such as "V3.5", "X3.5", "X29T", "X4.0", "V4.0"). We are instead
! just using a link-time symbol (see below).
!
! ITEM[0,0,16,0] = 8;
! ITEM[0,16,16,0] = SYI$_VERSION;
! ITEM[1,0,32,0] = VERSION_BUFFER;
! ITEM[2,0,32,0] = LEN;
! CH$FILL(0, 12, ITEM[3, A_]);
! STATUS = LIB$GET_EF(EVNT_FLAG);
! IF NOT .STATUS THEN EVNT_FLAG = 0;
! STATUS = $GETSYI(EFN=.EVNT_FLAG, ITMLST=ITEM);
! IF .STATUS
! THEN

    ! Version 3 systems will return "V3.x" in VERSION_BUFFER.
    !
    ! DBG$GV_CONTROL[DBG$V_CONTROL_VERSION_4] = NOT
    !     ((.VERSION_BUFFER[0] EQL 'V') AND (.VERSION_BUFFER[1] EQL '3'))

! ELSE

    ! $GETSYI failed. Make a guess that we are a 3B system.
    !
    ! DBG$GV_CONTROL[DBG$V_CONTROL_VERSION_4] = 1;

! LIB$FREE_EF(EVNT_FLAG);


! Initialize the bit that says whether we are on a 3B system.
! We rely on a link-time symbol DBG$GL_3B_SYSTEM.
!
DBG$GV_CONTROL[DBG$V_CONTROL_VERSION_4] = DBG$GL_3B_SYSTEM;


! Initialize the global which says whether we are trying to do
! keypad input.
!
DBG$GB_KEYPAD_INPUT = .DBG$GV_CONTROL[DBG$V_CONTROL_VERSION_4];


! Open the input device for reading.  If the OPENs and CONNECTs cannot be
! done successfully for logical devices 'DBG$INPUT' and 'DBG$OUTPUT', then
! try 'SYS$INPUT' and 'SYS$OUTPUT'.  If these fail, signal an error.  This
! causes a return to the command line interpreter in the operating system.
!
DBG$GL_INPFAB [FAB$L_FNA] = DBG_INPUT_DEVICE;
DBG$GL_INPFAB [FAB$B_FNS] = DBG_INP_DEV_SIZE;
OPEN_STATUS = $OPEN (FAB = DBG$GL_INPFAB);
IF NOT .OPEN_STATUS
THEN
    BEGIN
    DBG$GL_INPFAB [FAB$L_FNA] = SYS_INPUT_DEVICE;
    DBG$GL_INPFAB [FAB$B_FNS] = SYS_INP_DEV_SIZE;
    STATUS = $OPEN (FAB = DBG$GL_INPFAB);
    IF NOT .STATUS THEN $EXIT(CODE = .STATUS OR FATAL_BIT);
```

```
        END;


! Connect the input file.
!
DBG$GL_INPRAB[RAB$L_FAB] = DBG$GL_INPFAB;
STATUS = $CONNECT(RAB = DBG$GL_INPRAB);
IF NOT .STATUS THEN $EXIT(CODE = .STATUS OR FATAL_BIT);


! CREATE and OPEN the output file.
!
DBG$GL_OUTPFAB [FAB$L_FNA] = DBG_OUTPUT_DEVICE;
DBG$GL_OUTPFAB [FAB$B_FNS] = DBG_OUT_DEV_SIZE;
OUTPUT_STATUS = $CREATE (FAB = DBG$GL_OUTPFAB);
IF NOT .OUTPUT_STATUS
THEN
    BEGIN
    DBG$GL_OUTPFAB [FAB$L_FNA] = SYS_OUTPUT_DEVICE;
    DBG$GL_OUTPFAB [FAB$B_FNS] = SYS_OUT_DEV_SIZE;
    STATUS = $CREATE (FAB = DBG$GL_OUTPFAB);
    IF NOT .STATUS THEN $EXIT(CODE = .STATUS OR FATAL_BIT);
    END;


! CONNECT the output file.
!
DBG$GL_OUTPRAB[RAB$L_FAB] = DBG$GL_OUTPFAB;
STATUS = $CONNECT(RAB = DBG$GL_OUTPRAB);
IF NOT .STATUS THEN $EXIT(CODE = .STATUS OR FATAL_BIT);


! We need to delay this message output till SYS$INPUT, SYS$OUTPUT are
! established. Otherwise, DBG$PUTMSG in DBG$FINAL_HANDL does not know
! where to output the message.
!
IF NOT .OPEN_STATUS THEN SIGNAL(DBG$_UNAOPEDBGI, 0, .OPEN_STATUS);
IF NOT .OUTPUT_STATUS THEN SIGNAL(DBG$_UNACREDBGO, 0, .OUTPUT_STATUS);


! Get the terminal width.
!
DEVCHAR = DBG$GL_OUTPFAB[FAB$L_DEV];
IF .DEVCHAR[DEV$V_TRM]
THEN
    BEGIN
    LOCAL
        DEV_DESC: VECTOR[2,LONG],
        INFO_4: VECTOR[4,LONG],
        RETURN_LENGTH;

    DEV_DESC[0] = %X'010E0000' OR .DBG$GL_OUTPFAB [FAB$B_FNS];
    DEV_DESC[1] = .DBG$GL_OUTPFAB [FAB$L_FNA];
    INFO_4[0] = DVI$_DEVBUFSIZ^16 OR 4;
    INFO_4[1] = DBG$SRC_TERM_WIDTH;
    INFO_4[2] = RETURN_LENGTH;
    INFO_4[3] = 0;
```

```
	    STATUS = $GETDVI(DEVNAM=DEV_DESC, ITMLST=INFO_4);
	    IF NOT .STATUS THEN SIGNAL(.STATUS);
	    END

	ELSE
	    DBG$SRC_TERM_WIDTH = 80;


	! Set the flag that says resignal all exceptions from the user
	! program except for user-set breakpoints and tracepoints.
	!
	DBG$GB_RESIGNAL = TRUE;


	! Initialize the define settings.
	!
	DBG$SET_DEFINE_DEF();


	! Initialize the DEFINE symbol table by allocating space for
	! the header blocks, and initializing the fields to be zero.
	! Note that this must be done before DBG$SET_LANG.
	!
	HEADER = DBG$GET_MEMORY (DBG$K_DEFINE_HEADER_SIZE_W);
	HEADER [DEF$A_NEXT_LINK] = 0;
	HEADER [DEF$A_PREV_LINK] = 0;
	HEADER [DEF$A_DEFINE_LIST] = 0;
	DBG$GL_GLOBAL_DEFINE_PTR = .HEADER;
	HEADER = DBG$GET_MEMORY (DBG$K_DEFINE_HEADER_SIZE_W);
	HEADER [DEF$A_NEXT_LINK] = 0;
	HEADER [DEF$A_PREV_LINK] = 0;
	HEADER [DEF$A_DEFINE_LIST] = 0;
	DBG$GL_LOCAL_DEFINE_PTR = .HEADER;


	! Set the default language, namely MACRO.
	!
	DBG$SET_LANG(0,DBG$K_MACRO);


	! Set all the single command context bits to FALSE.  These bits refer to
	! context that is valid only during a single command, not across multiple
	! commands.
	!
	DBG$SET_CONTEXT ();


	! Initialize the new eventpoint data structures.
	!
	DBG$EVENT_INITIALIZATION ();


	! Initialize the Command Input Stream to DBG$INPUT
	!
	DBG$GL_CISHEAD = DBG$GET_MEMORY ((CIS_ELEMENTS+3)/%UPVAL);
	DBG$GL_CISHEAD[CIS$A_NEXT_LINK] = 0;
	DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] = CIS_DBG$INPUT;
```

```
;   2   DBG$GL_CISHEAD[CIS$A_INPUT_PTR] = DBG$GL_INPRAB;
;   2
;   2
;   2   ! Initialize the OUTPUT configuration
;   2   !
;   2   DBG$SET_OUT_DEF();
;   2
;   2
;   2   ! Note - processing initialization files must be done last.  If there was
;   2   ! an initialization file, add it to the command input stream. For DEBUG,
;   2   ! the initialization file is specified by the logical name DBG$INIT, and
;   2   ! for SUPERDEBUG, it is specified by the logical name SDBG$INIT.
;   2   !
;   2   DUMMY[0] = %X'010E0000'+256;
;   2   DUMMY[1] = DUMMY_BUFFER;
;   2
;   2
;   2   ! We need to allocate space for the file name and copy 'DBG$INIT' or
;   2   ! 'SDBG$INIT' into this space.  The reason for this is that DBG$CIS_REMOVE
;   2   ! will free up the space.  Also, fill in the string descriptor to
;   2   ! be used in SYS$TRNLOG.  Note - do NOT replace this with a %ASCID
;   2   ! declaration.  %ASCID causes the code to be non-shareable and thus
;   2   ! degrades performance.
;   2   !
;   2   DBG$GL_IND_COM_FILE = DBG$GET_MEMORY (3);
;   2   SDBGINIT_STGDESC [DSC$B_CLASS] = DSC$K_CLASS_S;
;   2   SDBGINIT_STGDESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
;   2   SDBGINIT_STGDESC [DSC$A_POINTER] = SDBGINIT_STG;
;   2   IF .DBG$GV_CONTROL[DBG$V_CONTROL_SDBG]
;   2   THEN
;   3       BEGIN
;   3       SDBGINIT_STGDESC [DSC$W_LENGTH] = 9;
;   3       CH$MOVE(9, UPLIT BYTE(%ASCII 'SDBG$INIT'), SDBGINIT_STG);
;   3       STATUS = $YS$TRNLOG(SDBGINIT_STGDESC, 0, DUMMY, 0, 0, 0);
;   3       IF .STATUS EQL SS$_NORMAL
;   3       THEN
;   4           BEGIN
;   4           DBG$GL_IND_COM_FILE[0] = 9;
;   4           CH$MOVE(9, UPLIT BYTE(%ASCII 'SDBG$INIT'), DBG$GL_IND_COM_FILE[1]);
;   4           DBG$CIS_CONNECTICF(FALSE);
;   3           END;
;   3
;   3       END
;   3
;   2   ELSE
;   3       BEGIN
;   3       SDBGINIT_STGDESC [DSC$W_LENGTH] = 8;
;   3       CH$MOVE(8, UPLIT BYTE(%ASCII 'DBG$INIT'), SDBGINIT_STG);
;   3       STATUS = $YS$TRNLOG(SDBGINIT_STGDESC, 0, DUMMY, 0, 0, 0);
;   3       IF .STATUS EQL SS$_NORMAL
;   3       THEN
;   4           BEGIN
;   4           DBG$GL_IND_COM_FILE[0] = 8;
;   4           CH$MOVE(8, UPLIT BYTE(%ASCII 'DBG$INIT'), DBG$GL_IND_COM_FILE[1]);
;   4           DBG$CIS_CONNECTICF(FALSE);
;   3           END;
;   3
;  853        0980  2          END;
;  854        0981  2
;  855        0982  2
;  856        0983  2       ! Initialization is complete and successful.  Output the DEBUG header
;  857        0984  2       ! message with the version number and return.
;  858        0985  2       !
;  859        0986  2       IF .DBG$GV_CONTROL[DBG$V_CONTROL_SDBG]
;  860        0987  2       THEN
;  861        0988  2           $FAO_TT_OUT('!/  VAX SUPERDEBUG Version 4.0-8!/')
;  862        0989  2
;  863        0990  2       ELSE
;  864        0991  2           $FAO_TT_OUT('!/  VAX DEBUG Version 4.0-8!/');
;  865        0992  2
;  866        0993  2       RETURN;
;  867        0994  1       END;



                                                                   .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                    54  55  50  4E  49  24  47  42  44  00004 P.AAB:  .ASCII  \DBG$INPUT\
                54  55  50  54  55  4F  24  47  42  44  0000D P.AAC:  .ASCII  \DBG$OUTPUT\
                    54  55  50  4E  49  24  53  59  53  00017 P.AAD:  .ASCII  \SYS$INPUT\
                54  55  50  54  55  4F  24  53  59  53  00020 P.AAE:  .ASCII  \SYS$OUTPUT\
                    54  49  4E  49  24  47  42  44  53  0002A P.AAF:  .ASCII  \SDBG$INIT\
                    54  49  4E  49  24  47  42  44  53  00033 P.AAG:  .ASCII  \SDBG$INIT\
                        54  49  4E  49  24  47  42  44  0003C P.AAH:  .ASCII  \DBG$INIT\
                        54  49  4E  49  24  47  42  44  00044 P.AAI:  .ASCII  \DBG$INIT\
                                                        22  0004C P.AAJ:  .BYTE   34
45  44  52  45  50  55  53  20  58  41  56  20  09  2F  21  0004D         .ASCII  \!/\<9>\ VAX SUPERDEBUG Version 4.0-8!/\
30  2E  34  20  6E  6F  69  73  72  65  56  20  47  55  42  0005C
                                            2F  21  38  2D  0006B
                                                        1D  0006F P.AAK:  .BYTE   29
56  20  47  55  42  45  44  20  58  41  56  20  09  2F  21  00070         .ASCII  \!/\<9>\ VAX DEBUG Version 4.0-8!/\
    2F  21  38  2D  30  2E  34  20  6E  6F  69  73  72  65  0007F

                                                          DBG_INPUT_DEVICE=   P.AAB
                                                          DBG_INP_DEV_SIZE=   9
                                                          DBG_OUTPUT_DEVICE=  P.AAC
                                                          DBG_OUT_DEV_SIZE=   10
                                                          SYS_INPUT_DEVICE=   P.AAD
                                                          SYS_INP_DEV_SIZE=   9
                                                          SYS_OUTPUT_DEVICE=  P.AAE
                                                          SYS_OUT_DEV_SIZE=   10
                                                                  .EXTRN  SYS$OPEN, SYS$EXIT
                                                                  .EXTRN  SYS$CONNECT, SYS$CREATE
                                                                  .EXTRN  SYS$GETDVI

                                                                  .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                          0FFC 00000              .ENTRY  DBG$INIT_DEBUG, Save R2,R3,R4,R5,R6,R7,R8,-  ; 0695
                                                                          R9,R10,R11
                          5B 00000000G  00  9E 00002              MOVAB   SYS$EXIT, R11
                          5A 00000000G  00  9E 00009              MOVAB   DBG$GV_CONTROL, R10
                          59 00000000G  00  9E 00010              MOVAB   DBG$GL_INPFAB, R9
                          58 00000000G  00  9E 00017              MOVAB   DBG$GL_OUTPFAB+44, R8
                          57 00000000'  EF  9E 0001E              MOVAB   DBG_INPUT_DEVICE, R7
```

```
                                                                      J 2
DBGLEVEL1                                                     16-Sep-1984 01:27:02    VAX-11 Bliss-32 V4.0-742                    Page  25
V04-000                                                       14-Sep-1984 12:17:02    [DEBUG.SRC]DBGLEVEL1.B32;1                        (7)

                                      5E       FE9C  CE  9E 00025         MOVAB   -356(SP), SP                                    :
                          00000000G   00             00  FB 0002A         CALLS   #0, DBG$INIT_MEMORY                             : 0746
        01   AA        01             04 00000000G   8F  F0 00031         INSV    #DBG$GL_3B_SYSTEM, #4, #1, DBG$GV_CONTROL+1     : 0785
             50     01         AA     01             04  EF 0003B         EXTZV   #4, #1, DBG$GV_CONTROL+1, R0                    : 0791
                          00000000G   00             50  90 00041         MOVB    R0, DBG$GB_KEYPAD_INPUT                         :
                                 2C   A9             67  9E 00048         MOVAB   DBG_INPUT_DEVICE, DBG$GL_INPFAB+44              : 0799
                                 34   A9             09  90 0004C         MOVB    #9, DBG$GL_INPFAB+52                            : 0800
                                      59             59  DD 00050         PUSHL   R9                                              : 0801
                          00000000G   00             01  FB 00052         CALLS   #1, SYS$OPEN                                    :
                                      53             50  D0 00059         MOVL    R0, OPEN_STATUS                                 :
                                      1F             53  E8 0005C         BLBS    OPEN_STATUS, 1$                                 : 0802
                                 2C   A9         13  A7  9E 0005F         MOVAB   SYS_INPUT_DEVICE, DBG$GL_INPFAB+44              : 0805
                                 34   A9             09  90 00064         MOVB    #9, DBG$GL_INPFAB+52                            : 0806
                                                     59  DD 00068         PUSHL   R9                                              : 0807
                          00000000G   00             01  FB 0006A         CALLS   #1, SYS$OPEN                                    :
                                      56             50  D0 00071         MOVL    R0, STATUS                                      :
                                      07             56  E8 00074         BLBS    STATUS, 1$                                      : 0808
                       7E             56             04  C9 00077         BISL3   #4, STATUS, -(SP)                               :
                                      6B             01  FB 0007B         CALLS   #1, SYS$EXIT                                    :
                          00000000G   00             69  9E 0007E  1$:    MOVAB   DBG$GL_INPFAB, DBG$GL_INPRAB+60                 : 0814
                                        00000000G    00  9F 00085         PUSHAB  DBG$GL_INPRAB                                   : 0815
                          00000000G   00             01  FB 0008B         CALLS   #1, SYS$CONNECT                                 :
                                      56             50  D0 00092         MOVL    R0, STATUS                                      :
                                      07             56  E8 00095         BLBS    STATUS, 2$                                      : 0816
                       7E             56             04  C9 00098         BISL3   #4, STATUS, -(SP)                               :
                                      6B             01  FB 0009C         CALLS   #1, SYS$EXIT                                    :
                                      68         09  A7  9E 0009F  2$:    MOVAB   DBG_OUTPUT_DEVICE, DBG$GL_OUTPFAB+44            : 0821
                                 08   A8             0A  90 000A3         MOVB    #10, DBG$GL_OUTPFAB+52                          : 0822
                                                 D4  A8  9F 000A7         PUSHAB  DBG$GL_OUTPFAB                                  : 0823
                          00000000G   00             01  FB 000AA         CALLS   #1, SYS$CREATE                                  :
                                      52             50  D0 000B1         MOVL    R0, OUTPUT_STATUS                               :
                                      1F             52  E8 000B4         BLBS    OUTPUT_STATUS, 3$                               : 0824
                                      68         1C  A7  9E 000B7         MOVAB   SYS_OUTPUT_DEVICE, DBG$GL_OUTPFAB+44            : 0827
                                 08   A8             0A  90 000BB         MOVB    #10, DBG$GL_OUTPFAB+52                          : 0828
                                                 D4  A8  9F 000BF         PUSHAB  DBG$GL_OUTPFAB                                  : 0829
                          00000000G   00             01  FB 000C2         CALLS   #1, SYS$CREATE                                  :
                                      56             50  D0 000C9         MOVL    R0, STATUS                                      :
                                      07             56  E8 000CC         BLBS    STATUS, 3$                                      : 0830
                       7E             56             04  C9 000CF         BISL3   #4, STATUS, -(SP)                               :
                                      6B             01  FB 000D3         CALLS   #1, SYS$EXIT                                    :
                          00000000G   00         D4  A8  9E 000D6  3$:    MOVAB   DBG$GL_OUTPFAB, DBG$GL_OUTPRAB+60               : 0836
                                        00000000G    00  9F 000DE         PUSHAB  DBG$GL_OUTPRAB                                  : 0837
                          00000000G   00             01  FB 000E4         CALLS   #1, SYS$CONNECT                                 :
                                      56             50  D0 000EB         MOVL    R0, STATUS                                      :
                                      07             56  E8 000EE         BLBS    STATUS, 4$                                      : 0838
                       7E             56             04  C9 000F1         BISL3   #4, STATUS, -(SP)                               :
                                      6B             01  FB 000F5         CALLS   #1, SYS$EXIT                                    :
                                      11             53  E8 000F8  4$:    BLBS    OPEN_STATUS, 5$                                 : 0845
                                                     53  DD 000FB         PUSHL   OPEN_STATUS                                     :
                                                     7E  D4 000FD         CLRL    -(SP)                                           :
                                        00028FDB     8F  DD 000FF         PUSHL   #167899                                         :
                          00000000G   00             03  FB 00105         CALLS   #3, LIB$SIGNAL                                  :
                                      11             52  E8 0010C  5$:    BLBS    OUTPUT_STATUS, 6$                               : 0846
                                                     52  DD 0010F         PUSHL   OUTPUT_STATUS                                   :
                                                     7E  D4 00111         CLRL    -(SP)                                           :
                                        00028FE3     8F  DD 00113         PUSHL   #167907                                         :
                          00000000G   00             03  FB 00119         CALLS   #3, LIB$SIGNAL                                  :
```

```
                                                   K  2
DBGLEVEL1                                          16-Sep-1984 01:27:02    VAX-11 Bliss-32 V4.0-742            Page  26
V04-000                                            14-Sep-1984 12:17:02    [DEBUG.SRC]DBGLEVEL1.B32;1                 (7)

                      50         14 AB 9E 00120 6$:   MOVAB   DBG$GL_OUTPFAB+64, DEVCHAR                 ; 0851
        4C            60            02 E1 00124       BBC     #2, (DEVCHAR), 7$                          ; 0852
                      50         08 A8 9A 00128       MOVZBL  DBG$GL_OUTPFAB+52, R0                      ; 0860
14      AE            50   010E0000 8F C9 0012C       BISL3   #17694720, R0, DEV_DESC                    ;
                   18 AE            68 D0 00135       MOVL    DBG$GL_OUTPFAB+44, DEV_DESC+4              ; 0861
                   04 AE   00080004 8F D0 00139       MOVL    #524292, INFO_4                            ; 0862
                   08 AE  00000000G 00 9E 00141       MOVAB   DBG$SRC_TERM_WIDTH, INFO_4+4               ; 0863
                   0C AE            6E 9E 00149       MOVAB   RETURN_LENGTH, INFO_4+8                    ; 0864
                                 10 AE D4 0014D       CLRL    INFO_4+12                                  ; 0865
                                    7E 7C 00150       CLRQ    -(SP)                                      ; 0866
                                    7E 7C 00152       CLRQ    -(SP)                                      ;
                                 14 AE 9F 00154       PUSHAB  INFO_4                                     ;
                                 28 AE 9F 00157       PUSHAB  DEV_DESC                                   ;
                                    7E 7C 0015A       CLRQ    -(SP)                                      ;
            00000000G 00            08 FB 0015C       CALLS   #8, SYS$GETDVI                             ;
                      56            50 D0 00163       MOVL    R0, STATUS                                 ;
                      13            56 E8 00166       BLBS    STATUS, 8$                                 ; 0867
                                    56 DD 00169       PUSHL   STATUS                                     ;
            00000000G 00            01 FB 0016B       CALLS   #1, LIB$SIGNAL                             ;
                                    08 11 00172       BRB     8$                                         ; 0852
            00000000G 00         50 8F 9A 00174 7$:   MOVZBL  #80, DBG$SRC_TERM_WIDTH                    ; 0871
            00000000G 00            01 90 0017C 8$:   MOVB    #1, DBG$GB_RESIGNAL                        ; 0877
            00000000G 00            00 FB 00183       CALLS   #0, DBG$SET_DEFINE_DEF                     ; 0882
                                    03 DD 0018A       PUSHL   #3                                         ; 0889
            00000000G 00            01 FB 0018C       CALLS   #1, DBG$GET_MEMORY                         ;
                                    60 7C 00193       CLRQ    (HEADER)                                   ; 0890
                                 08 A0 D4 00195       CLRL    8(HEADER)                                  ; 0892
            00000000G 00            50 D0 00198       MOVL    HEADER, DBG$GL_GLOBAL_DEFINE_PTR           ; 0893
                                    03 DD 0019F       PUSHL   #3                                         ; 0894
            00000000G 00            01 FB 001A1       CALLS   #1, DBG$GET_MEMORY                         ;
                                    60 7C 001A8       CLRQ    (HEADER)                                   ; 0895
                                 08 A0 D4 001AA       CLRL    8(HEADER)                                  ; 0897
            00000000G 00            50 D0 001AD       MOVL    HEADER, DBG$GL_LOCAL_DEFINE_PTR            ; 0898
                                    7E 7C 001B4       CLRQ    -(SP)                                      ; 0903
                0000V CF            02 FB 001B6       CALLS   #2, DBG$SET_LANG                           ;
                 FE34 CF            00 FB 001BB       CALLS   #0, DBG$SET_CONTEXT                        ; 0910
            00000000G 00            00 FB 001C0       CALLS   #0, DBG$EVENT_INITIALIZATION               ; 0915
                                    0E DD 001C7       PUSHL   #14                                        ; 0920
            00000000G 00            01 FB 001C9       CALLS   #1, DBG$GET_MEMORY                         ;
            00000000G 00            50 D0 001D0       MOVL    R0, DBG$GL_CISHEAD                         ;
                                 08 A0 D4 001D7       CLRL    8(R0)                                      ; 0921
                                 02 A0 94 001DA       CLRB    2(R0)                                      ; 0922
                   04 A0  00000000G 00 9E 001DD       MOVAB   DBG$GL_INPRAB, 4(R0)                       ; 0923
            00000000G 00            00 FB 001E5       CALLS   #0, DBG$SET_OUT_DEF                        ; 0928
                   F8 AD   010E0100 8F D0 001EC       MOVL    #17694976, DUMMY                           ; 0936
                   FC AD         5C AE 9E 001F4       MOVAB   DUMMY_BUFFER, DUMMY+4                      ; 0937
                                    03 DD 001F9       PUSHL   #3                                         ; 0947
            00000000G 00            01 FB 001FB       CALLS   #1, DBG$GET_MEMORY                         ;
            00000000G 00            50 D0 00202       MOVL    R0, DBG$GL_IND_COM_FILE                    ;
                   32 AE       010E 8F B0 00209       MOVW    #270, SDBGINIT_STGDESC+2                   ; 0949
                   34 AE         24 AE 9E 0020F       MOVAB   SDBGINIT_STG, SDBGINIT_STGDESC+4           ; 0950
        37            6A            01 E1 00214       BBC     #1, DBG$GV_CONTROL, 9$                     ; 0951
                   30 AE            09 B0 00218       MOVW    #9, SDBGINIT_STGDESC                       ; 0954
24      AE         26 A7            09 28 0021C       MOVC3   #9, P.AAF, SDBGINIT_STG                    ; 0955
                                    7E 7C 00222       CLRQ    -(SP)                                      ; 0956
                                    7E D4 00224       CLRL    -(SP)                                      ;
                                 F8 AD 9F 00226       PUSHAB  DUMMY                                      ;
```

```
                                          7E  D4 00229          CLRL    -(SP)                                   ;
                                    44    AE  9F 0022B          PUSHAB  SDBGINIT_STGDESC                        ;
                   00000000G  00          06  FB 0022E          CALLS   #6, SYS$TRNLOG                          ;
                              56          50  D0 00235          MOVL    R0, STATUS                              ;
                              01          56  D1 00238          CMPL    STATUS, #1                              ; 0957
                                          4E  12 0023B          BNEQ    11$                                     ;
                              50 00000000G 00  D0 0023D         MOVL    DBG$GL_IND_COM_FILE, R0                 ; 0960
                              60          09  90 00244          MOVB    #9, (R0)                                ;
          01    A0         2F  A7         09  28 00247          MOVC3   #9, P.AAG, 1(R0)                        ; 0961
                                          35  11 0024D          BRB     10$                                     ; 0962
                           30  AE         08  B0 0024F 9$:      MOVW    #8, SDBGINIT_STGDESC                    ; 0969
          24    AE         38  A7         08  28 00253          MOVC3   #8, P.AAH, SDBGINIT_STG                 ; 0970
                                          7E  7C 00259          CLRQ    -(SP)                                   ; 0971
                                          7E  D4 0025B          CLRL    -(SP)                                   ;
                                    F8    AD  9F 0025D          PUSHAB  DUMMY                                   ;
                                          7E  D4 00260          CLRL    -(SP)                                   ;
                                    44    AE  9F 00262          PUSHAB  SDBGINIT_STGDESC                        ;
                   00000000G  00          06  FB 00265          CALLS   #6, SYS$TRNLOG                          ;
                              56          50  D0 0026C          MOVL    R0, STATUS                              ;
                              01          56  D1 0026F          CMPL    STATUS, #1                              ; 0972
                                          17  12 00272          BNEQ    11$                                     ;
                              50 00000000G 00  D0 00274         MOVL    DBG$GL_IND_COM_FILE, R0                 ; 0975
                              60          08  90 0027B          MOVB    #8, (R0)                                ;
          01    A0         40  A7         08  28 0027E          MOVC3   #8, P.AAI, 1(R0)                        ; 0976
                                          7E  D4 00284 10$:     CLRL    -(SP)                                   ; 0977
                       0000V  CF          01  FB 00286          CALLS   #1, DBG$CIS_CONNECTICF                  ;
                05            6A          01  E1 0028B 11$:     BBC     #1, DBG$GV_CONTROL, 12$                 ; 0986
                                    48    A7  9F 0028F          PUSHAB  P.AAJ                                   ; 0988
                                          03  11 00292          BRB     13$                                     ;
                                    6B    A7  9F 00294 12$:     PUSHAB  P.AAK                                   ; 0991
                   00000000G  00          01  FB 00297 13$:     CALLS   #1, DBG$FAO_OUT                         ;
                                              04 0029E          RET                                             ; 0994

; Routine Size:  671 bytes,    Routine Base:  DBG$CODE + 0224
;  869          0995  1 GLOBAL ROUTINE dbg$ins_opcodes (user_pc_value) : NOVALUE =
;  870          0996  2 BEGIN
;  871          0997  2 0
;  872          0998  1 END;




                                        0000 00000         .ENTRY  DBG$INS_OPCODES, Save nothing                   ; 0995
                                          04 00002         RET                                                     ; 0998

; Routine Size:  3 bytes,    Routine Base:  DBG$CODE + 04C3
```

```
GLOBAL ROUTINE dbg$conv_r_50( OPWORD, DST_PTR ) =
!++
! FUNCTIONAL DESCRIPTION:
!       THIS ROUTINE IS A SPECIAL-PURPOSE RAD50-TO-ASCII CONVERSION
!       ROUTINE.  IT TAKES A LONGWORD CONTAINING 2 RAD50 WORDS,
!       CONVERTS THEM TO A 6-character STRING, AND 'STUFFS' THESE
!       characters INTO THE BYTE VECTOR WE ARE PASSED A POINTER TO.
!
! WARNING:
!       THE REST OF THE CODE WHICH INTERFACES TO THIS USES THE
!       LITERAL 'OP_CH_SIZE' TO REFER TO THIS 6-character STRING,
!       BUT THIS SIZE IS IMPLICIT IN THE CODE HERE.  IF THIS
!       CODE IS CHANGED, THE LITERAL MUST ALSO BE CHANGED.  JUST
!       CHANGING THE LITERAL IS NOT ENOUGH.
!
! CALLING SEQUENCE:
!       dbg$conv_r_50 ( LONGWORD, BYTE_POINTER );
!
! INPUTS:
!       OPWORD  - THE LONGWORD WHICH CONTAINS THE 6 RAD50 characters.
!               - THIS WORD COMES FROM THE OP_NAME FIELD OF THE
!                 dbg$GB_OPINFO DATA VECTOR.
!       DST_PTR - ANY BYTE ADDRESS INTO WHICH THIS ROUTINE WILL
!                 STUFF THE 6 ASCII characters OBTAINED FROM OPWORD.
!
! IMPLICIT INPUTS:
!       OP_CH_SIZE - FROM VAXOPS.REQ, WHICH MUST BE 6 TO MATCH THIS CODE.
!
! OUTPUTS:
!       NONE.
!
! IMPLICIT OUTPUTS:
!       THE 6 ASCII chars ARE STUFFED BACK INTO THE USER-SUPPLIED
!       STRING.
!
! ROUTINE VALUE:
!       The number of non-blank characters stuffed into
!       the output string.
!
! SIDE EFFECTS:
!       SEE IMPLICIT OUTPUTS.
!--

    BEGIN

    MAP
        DST_PTR : REF VECTOR[,BYTE];     ! WHERE TO STUFF THE chars.

    LOCAL
        non_blanks,
        J,              ! INDEX.
        W,              ! THE LONGWORD.
        PTR : REF VECTOR[,BYTE];

    BIND
        DIVTAB = UPLIT(1, %O'50', %O'3100') : VECTOR;
```

```
;  931        1056  2          ! JUST EXTRACT EACH OF THE TWO WORDS, CONVERT THEM,
;  932        1057  2          ! AND STUFF BACK THE RESTULS.
;  933        1058  2
;  934        1059  2          PTR = .DST_PTR;
;  935        1060  2          non_blanks = 0;
;  936        1061  2
;  937        1062  2          INCR K FROM 0 TO 16 BY 16
;  938        1063  2              DO
;  939        1064  3              BEGIN ! DO THE CONVERSION ON BOTH WORDS SEPARATELY.
;  940        1065  3
;  941        1066  3              W = .OPWORD<.K,16>;
;  942        1067  3
;  943        1068  3              DECR I FROM 2 TO 0 DO
;  944        1069  4                  BEGIN
;  945        1070  4                  J = .W/.DIVTAB[.I]; W = .W - .J*.DIVTAB[.I];
;  946        1071  4                  IF .J NEQ 0
;  947        1072  4                  THEN
;  948        1073  5                      BEGIN
;  949        1074  5                      IF .J NEQ %O'33'
;  950        1075  5                      THEN
;  951        1076  6                          BEGIN
;  952        1077  6                          IF .J LSS %O'33' THEN J = .J + %O'56';
;  953        1078  6                          J = .J + %O'11';
;  954        1079  5                          END;
;  955        1080  5                      J = .J + %O'11';
;  956        1081  5                      END
;  957        1082  4                  ELSE
;  958        1083  4                      J = %O'40';
;  959        1084  4
;  960        1085  4                  ! AT THIS POINT, A SINGLE char IS IN BYTE 0 OF J.
;  961        1086  4                  ! Stuff the character back and tally up the
;  962        1087  4                  ! number of non-blank ones.
;  963        1088  4
;  964        1089  5                  IF( ((.PTR)<0,8> = .J) NEQ %O'40' )
;  965        1090  4                  then
;  966        1091  4                          non_blanks = .non_blanks +1;
;  967        1092  4
;  968        1093  4                  PTR = .PTR + 1;
;  969        1094  3                  END;
;  970        1095  3
;  971        1096  2          END;
;  972        1097  2          ! END OF K LOOP.
;  973        1098  2
;  974        1099  2      !Return the number of non-blank characters
;  975        1100  2      ! we stuffed back.
;  976        1101  2
;  977        1102  2      RETURN(.non_blanks);
;  978        1103  1      END; ! OF conv_r_50 ROUTINE.



                                                                  .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                                                      0008D               .BLKB   3
           00000640  00000028  00000001               00090  P.AAL:       .LONG   1, 40, 1600                         ;

                                                             DIVTAL=              P.AAL
```

```
                                                              .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                         00FC 00000           .ENTRY  DBG$CONV_R_50, Save R2,R3,R4,R5,R6,R7        ; 0999
                         57 00000000'  EF  9E 00002           MOVAB   DIVTAB, R7                                   ;
                         54        08  AC  D0 00009           MOVL    DST_PTR, PTR                                 ; 1059
                                       56  D4 0000D           CLRL    NON_BLANKS                                   ; 1060
                                       53  D4 0000F           CLRL    K                                            ; 1062
      55      04   AC    10            53  EF 00011 1$:       EXTZV   K, #16, OPWORD, W                            ; 1066
                         51            02  D0 00017           MOVL    #2, I                                        ; 1068
                   50    55          6741  C7 0001A 2$:       DIVL3   DIVTAB[I], W, J                              ; 1070
                   52    50          6741  C5 0001F           MULL3   DIVTAB[I], J, R2                             ;
                         55            52  C2 00024           SUBL2   R2, W                                        ;
                                       50  D5 00027           TSTL    J                                            ; 1071
                                       12  13 00029           BEQL    5$                                           ;
                         1B            50  D1 0002B           CMPL    J, #27                                       ; 1074
                                       08  13 0002E           BEQL    4$                                           ;
                                       03  18 00030           BGEQ    3$                                           ; 1077
                         50            2E  C0 00032           ADDL2   #46, J                                       ;
                         50            09  C0 00035 3$:       ADDL2   #9, J                                        ; 1078
                         50            09  C0 00038 4$:       ADDL2   #9, J                                        ; 1080
                                       03  11 0003B           BRB     6$                                           ; 1071
                         50            20  D0 0003D 5$:       MOVL    #32, J                                       ; 1083
                         64            50  90 00040 6$:       MOVB    J, (PTR)                                     ; 1089
                         20            50  D1 00043           CMPL    J, #32                                       ;
                                       02  13 00046           BEQL    7$                                           ;
                                       56  D6 00048           INCL    NON_BLANKS                                   ; 1091
                                       54  D6 0004A 7$:       INCL    PTR                                          ; 1093
                         CB            51  F4 0004C           SOBGEQ  I, 2$                                        ; 1068
    FFBC           53    10            10  F1 0004F           ACBL    #16, #16, K, 1$                              ; 1062
                         50            56  D0 00055           MOVL    NON_BLANKS, R0                               ; 1102
                                           04 00058           RET                                                  ; 1103

; Routine Size:  89 bytes,    Routine Base:  DBG$CODE + 04C6
```

```
global routine dbg$octal_valtostg_cvt(value, count) =

!  ++
!  Functional Description:
!        Convert a value to an ascii string. The string, when
!        printed, displays the octal representation of the value.
!
!  Inputs
!        value - the actual value we are to convert.
!        count - the number of characters in the result string.
!
!  Routine Value
!        A pointer to a counted string.
!  Side effects
!        Allocates space for the result. This should be
!        freed by the caller when he is finished with it.
!  --

    begin

    own
        result_ptr : ref vector[,byte],

        tran_table : vector[8,byte]
                     initial( byte( %ascii '01234567'));

    ! allocate space for the string.
    result_ptr = dbg$get_memory(((1+.count)/%upval)+1);

    if .result_ptr eql 0 then signal(dbg$_nofree)
    else

        ! fill in result string from right to left.
        decr i from .count to 1 do
            begin
            result_ptr[.i] = .tran_table[.value mod 8];
            value = .value / 8
            end;

    ! fill in the count.
    result_ptr[0] = .count;

    .result_ptr

    end; ! of dbg$octal_valtostg_cvt


                                                  .PSECT  DBG$OWN,NOEXE,  PIC,2

                                       00000 RESULT_PTR:
                                                  .BLKB   4
37  36  35  34  33  32  31  30         00004 TRAN_TABLE:
                                                  .ASCII  \01234567\                          ;
```

```
                                                                           .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                                        000C 00000         .ENTRY  DBG$OCTAL_VALTOSTG_CVT, Save R2,R3          ; 1104
                                        53 00000000'   EF  9E 00002         MOVAB   RESULT_PTR, R3                              ;
                          50       08   AC            01  C1 00009         ADDL3   #1, COUNT, R0                               ; 1131
                                        50            04  C6 0000E         DIVL2   #4, R0                                      ;
                                                 01   A0  9F 00011         PUSHAB  1(R0)                                       ;
                            00000000G   00            01  FB 00014         CALLS   #1, DBG$GET_MEMORY                          ;
                                        63            50  D0 0001B         MOVL    R0, RESULT_PTR                              ;
                                        52            63  D0 0001E         MOVL    RESULT_PTR, R2                              ; 1133
                                                      0F  12 00021         BNEQ    1$                                          ;
                                           00028332   8F  DD 00023         PUSHL   #164658                                     ;
                            00000000G   00            01  FB 00029         CALLS   #1, LIB$SIGNAL                              ;
                                                      1F  11 00030         BRB     4$                                          ;
                          51       08   AC            01  C1 00032 1$:     ADDL3   #1, COUNT, I                                ; 1137
                                                      15  11 00037         BRB     3$                                          ;
           7E             00       04   AC            01  7A 00039 2$:     EMUL    #1, VALUE, #0, -(SP)                        ; 1139
           50             50            8E            08  7B 0003F         EDIV    #8, (SP)+, R0, R0                           ;
                                      6142       04 A340  90 00044         MOVB    TRAN_TABLE[R0], (I)[R2]                     ;
                                   04   AC            08  C6 0004A         DIVL2   #8, VALUE                                   ; 1140
                                        E8            51  F5 0004E 3$:     SOBGTR  I, 2$                                       ;
                                   00   B3       08   AC  90 00051 4$:     MOVB    COUNT, @RESULT_PTR                          ; 1144
                                        50            63  D0 00056         MOVL    RESULT_PTR, R0                              ; 1148
                                                          04 00059         RET                                                 ;

; Routine Size:  90 bytes,    Routine Base:  DBG$CODE + 051F


; 1025          1149  1
```

```
1 global routine dbg$decimal_valtostg_cvt(in_value_ptr,len) =
1 !++
1 ! Functional description:
1 !       Converts a value to an ascii string to be printed.
1 ! Inputs
1 !       in_value_ptr -  points to the place in memory where the
1 !                       value is stored.
1 !       len          -  length in bytes of the value
1 ! Routine value
1 !       A pointer to a counted string with the result.
1 !       Storage for the result string is allocated dynamically
1 !       by calling dbg$get_memory.
1 ! Side effects
1 !       Allocates space for the result. This should be
1 !       released by the caller.
1 !--
2     begin
2     map
2         in_value_ptr : ref bitvector [] ;
2
2     local
2         value_copy : bitvector[128],
2         value_ptr : ref bitvector[],
2         sign_flag,
2         string : ref vector[,byte],
2         power_of_two,
2         new_string : ref vector[,byte],
2         new_power_of_two;
2
2
2                             ! define extended precision add routine which
2                             ! operates on counted ascii strings.
2                             routine addc(a,b) =
3                                 begin
3                                 map a : ref vector [,byte],
3                                     b : ref vector [,byte];
3                                 local
3                                     temp,
3                                     m,
3                                     n,
3                                     carry,
3                                     result : ref vector [,byte];
3                                 own ctable : vector[20,byte]
3                                     initial(byte('01234567890123456789'));
3                                 ! n is size of larger argument
3                                 n = (if .a[0] gtr .b[0] then .a[0] else .b[0]);
3                                 ! m is size of smaller argument.
3                                 m = (if .a[0] gtr .b[0] then .b[0] else .a[0]);
3                                 ! dbg$get_memory takes longword sizes.
3                                 result = dbg$get_memory((2+.n)/4+1);
3                                 carry = 0;
3                                 decr i from .n to 1+.n-.m do
4                                     begin
4                                     result[.i+1] = .ctable[temp =
5                                         (.carry+
6                                             (if .a[0] gtr .b[0]
6                                             then .a[.i] +
DBGLEVEL1                                        G 3
V04-000                                  16-Sep-1984 01:27:02   VAX-11 Bliss-32 V4.0-742          Page 35
                                         14-Sep-1984 12:17:02   [DEBUG.SRC]DBGLEVEL1.B32;1           (11)

; 1084        1207  6                            .b[.i-(.n-.m)]
; 1085        1208  6                    else .b[.i] +
; 1086        1209  6                            .a[.i-(.n-.m)])
; 1087        1210  4                            - 2*%C'0')];
; 1088        1211  4               carry = .temp geq 10;
; 1089        1212  3               end;
; 1090        1213  3           decr i from .n-.m to 1 do
; 1091        1214  4               begin
; 1092        1215  4               result[.i+1] = .ctable[temp=
; 1093        1216  5                    (.carry+
; 1094        1217  6                    (if .a[0] gtr .b[0]
; 1095        1218  6                    then .a[.i]
; 1096        1219  5                    else .b[.i]) -
; 1097        1220  4                    %C'0')];
; 1098        1221  4               carry = .temp geq 10;
; 1099        1222  3               end;
; 1100        1223  3           if .carry eql 1 then
; 1101        1224  4               begin
; 1102        1225  4               result[0] = .n+1;
; 1103        1226  4               result[1] = %C'1';
; 1104        1227  4               end
; 1105        1228  3           else
; 1106        1229  4               begin
; 1107        1230  4               result[0] = .n;
; 1108        1231  4               ch$move(.n,
; 1109        1232  4                   result[2],result[1]);
; 1110        1233  3               end;
; 1111        1234  3           .result
; 1112        1235  2           end; ! addc


                                                                .PSECT  DBG$OWN,NOEXE,  PIC,2

34  33  32  31  30  39  38  37  36  35  34  33  32  31  30  0000C CTABLE: .ASCII  \01234567890123456789\          ;
                                        39  38  37  36  35  0001B                                                 ;


                                                                .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                        07FC 00000 ADDC:   .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10    ; 1182
                       5A 00000000'  EF   9E 00002         MOVAB   CTABLE, R10                         ;
                                     58   D4 00009         CLRL    R8                                  ; 1195
                 08    BC      04    BC   91 0000B         CMPB    @A, @B                              ;
                                     08   1B 00010         BLEQU   1$                                  ;
                                     58   D6 00012         INCL    R8                                  ;
                       52      04    BC   9A 00014         MOVZBL  @A, N                               ;
                                     04   11 00018         BRB     2$                                  ;
                       52      08    BC   9A 0001A 1$:     MOVZBL  @B, N                               ;
                       06            58   E9 0001E 2$:     BLBC    R8, 3$                              ; 1197
                       54      08    BC   9A 00021         MOVZBL  @B, M                               ;
                                     04   11 00025         BRB     4$                                  ;
                       54      04    BC   9A 00027 3$:     MOVZBL  @A, M                               ;
                       50      02    A2   9E 0002B 4$:     MOVAB   2(R2), R0                           ; 1199
                       50            04   C6 0002F         DIVL2   #4, R0                              ;
                               01    A0   9F 00032         PUSHAB  1(R0)                               ;
```

```
         000000000G   00         01  FB 00035       CALLS   #1, DBG$GET_MEMORY          ;
                      56         50  D0 0003C       MOVL    R0, RESULT                  ;
                                 53  D4 0003F       CLRL    CARRY                       ; 1200
     50               52         54  C3 00041       SUBL3   M, N, R0                    ; 1201
                      57    01   A0  9E 00045       MOVAB   1(R0), R7                   ;
     55               52         54  C3 00049       SUBL3   M, N, R5                    ; 1207
                      50         52  D0 0004D       MOVL    N, I                        ;
                                 39  11 00050       BRB     9$                          ;
     51               50         55  C3 00052 5$:   SUBL3   R5, I, R1                   ;
                      0C         58  E9 00056       BLBC    R8, 6$                      ;
                      59    04 BC40  9A 00059       MOVZBL  @A[I], R9                   ;
                      51    08 BC41  9A 0005E       MOVZBL  @B[R1], R1                  ;
                                 0A  11 00063       BRB     7$                          ;
                      59    08 BC40  9A 00065 6$:   MOVZBL  @B[I], R9                   ; 1209
                      51    04 BC41  9A 0006A       MOVZBL  @A[R1], R1                  ;
                      51         59  C0 0006F 7$:   ADDL2   R9, R1                      ;
                      54    A0 A143  9E 00072       MOVAB   -96(R1)[CARRY], TEMP        ; 1204
                 01 A046       6A44  90 00077       MOVB    CTABLE[TEMP], 1(I)[RESULT]  ; 1203
                                 51  D4 0007D       CLRL    R1                          ; 1211
                      0A         54  D1 0007F       CMPL    TEMP, #10                   ;
                                 02  19 00082       BLSS    8$                          ;
                                 51  D6 00084       INCL    R1                          ;
                      53         51  D0 00086 8$:   MOVL    R1, CARRY                   ;
                                 50  D7 00089       DECL    I                           ; 1201
                      57         50  D1 0008B 9$:   CMPL    I, R7                       ;
                                 C2  18 0008E       BGEQ    5$                          ;
                      50    01   A5  9E 00090       MOVAB   1(R5), I                    ; 1213
                                 26  11 00094       BRB     14$                         ;
                      07         58  E9 00096 10$:  BLBC    R8, 11$                     ; 1217
                      51    04 BC40  9A 00099       MOVZBL  @A[I], R1                   ; 1218
                                 05  11 0009E       BRB     12$                         ;
                      51    08 BC40  9A 000A0 11$:  MOVZBL  @B[I], R1                   ; 1219
                      54    D0 A143  9E 000A5 12$:  MOVAB   -48(R1)[CARRY], TEMP        ; 1216
                 01 A046       6A44  90 000AA       MOVB    CTABLE[TEMP], 1(I)[RESULT]  ; 1215
                                 51  D4 000B0       CLRL    R1                          ; 1221
                      0A         54  D1 000B2       CMPL    TEMP, #10                   ;
                                 02  19 000B5       BLSS    13$                         ;
                                 51  D6 000B7       INCL    R1                          ;
                      53         51  D0 000B9 13$:  MOVL    R1, CARRY                   ;
                      D7         50  F5 000BC 14$:  SOBGTR  I, 10$                      ; 1213
                      01         53  D1 000BF       CMPL    CARRY, #1                   ; 1223
                                 0A  12 000C2       BNEQ    15$                         ;
     66               52         01  81 000C4       ADDB3   #1, N, (RESULT)             ; 1225
                 01   A6         31  90 000C8       MOVB    #49, 1(RESULT)              ; 1226
                                 09  11 000CC       BRB     16$                         ; 1223
                      66         52  90 000CE 15$:  MOVB    N, (RESULT)                 ; 1230
01   A6          02   A6         52  28 000D1       MOVC3   N, 2(RESULT), 1(RESULT)     ; 1232
                      50         56  D0 000D7 16$:  MOVL    RESULT, R0                  ; 1235
                                     04 000DA       RET                                 ;

; Routine Size:  219 bytes,    Routine Base:  DBG$CODE + 0579


; 1113          1236  2
; 1114          1237  2                 ! Copy the value to be examined into a local variable
; 1115          1238  2                 !
; 1116          1239  2                 value_ptr = value_copy;
```

```
2                       ch$move (16, .in_value_ptr, .value_ptr);
2
2                       ! now build up print string representing the
2                       ! octaword integer.
2                       string = dbg$get_memory(1);
2                       power_of_two = dbg$get_memory(1);
2                       (.string) = %ASCIC'0';
2                       (.power_of_two) = %ASCIC'1';
2                       sign_flag = 0;
2                       IF .value_ptr[8*.len-1] EQL 1
2                       THEN ! sign bit set.
3                           BEGIN
3                           sign_flag = 1;
3                           ! negate number.
3                           incr i from 0 to .len/4-1 do
3                               (.value_ptr+4*.i) =
3                                   NOT .(.value_ptr+4*.i);
2                           END;
2                       incr i from 0 to (8*.len)-2 do
3                           begin
3                           ! look at the ith bit of the integer.
3                           IF .value_ptr[.i] EQL 1
3                           THEN
4                               BEGIN
4                               ! if the bit is set, must add in the
4                               ! appropriate power of two.
4                               new_string = addc(.string,.power_of_two);
4                               dbg$rel_memory(.string);
4                               string = .new_string;
3                               END;
3                           new_power_of_two = addc(
3                               .power_of_two,.power_of_two);
3                           dbg$rel_memory(.power_of_two);
3                           power_of_two = .new_power_of_two;
2                           end; ! loop
2                       IF .sign_flag EQL 1
2                       THEN
3                           BEGIN
3                           ! un-complement number
3                           ! (we complemented earlier)
3                           incr i from 0 to .len/4-1 do
3                               (.value_ptr+4*.i) =
3                                   NOT .(.value_ptr+4*.i);
3                           ! add 1 (we complemented earlier)
3                           new_string = addc(.string,
3                               UPLIT BYTE(%ASCIC '1'));
3                           ! append minus sign.
3                           string = dbg$get_memory(1+(2+.new_string[0])/4);
3                           string[0] = 1+.new_string[0];
3                           string[1] = %C'-';
3                           ch$move(.new_string[0],new_string[1],string[2]);
3                           dbg$rel_memory(.new_string);
2                           END;
2   dbg$rel_memory(.power_of_two);
2   .string
1   end; ! of dbg$decimal_valtostg_cvt
```

```
                                                          .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                                   31  01  0009C P.AAM:   .ASCII  <1>\1\                                         ;


                                                          .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                     0FFC  00000          .ENTRY  DBG$DECIMAL_VALTOSTG_CVT, Save R2,R3,R4,R5,-; 1150
                                                                  R6,R7,R8,R9,R10,R11                         ;
                       5B 00000000G  00   9E 00002        MOVAB   DBG$REL_MEMORY, R11                        ;
                       5E            10   C2 00009        SUBL2   #16, SP                                    ;
                       57            6E   9E 0000C        MOVAB   VALUE_COPY, VALUE_PTR                      ; 1239
          67       04  BC            10   28 0000F        MOVC3   #16, @IN_VALUE_PTR, (VALUE_PTR)            ; 1240
                                     01   DD 00014        PUSHL   #1                                         ; 1244
              00000000G  00          01   FB 00016        CALLS   #1, DBG$GET_MEMORY                         ;
                       56            50   D0 0001D        MOVL    R0, STRING                                 ;
                                     01   DD 00020        PUSHL   #1                                         ; 1245
              00000000G  00          01   FB 00022        CALLS   #1, DBG$GET_MEMORY                         ;
                       59            50   D0 00029        MOVL    R0, POWER_OF_TWO                           ;
                       66      3001  8F   3C 0002C        MOVZWL  #12289, (STRING)                           ; 1246
                       69      3101  8F   3C 00031        MOVZWL  #12545, (POWER_OF_TWO)                     ; 1247
                                     55   D4 00036        CLRL    SIGN_FLAG                                  ; 1248
                       53        08  AC   D0 00038        MOVL    LEN, R3                                    ; 1249
          50           53            03   78 0003C        ASHL    #3, R3, R0                                 ;
                                     50   D7 00040        DECL    R0                                         ;
 51       67           01            50   EF 00042        EXTZV   R0, #1, (VALUE_PTR), R1                    ;
                       01            51   D1 00047        CMPL    R1, #1                                     ;
                                     15   12 0004A        BNEQ    3$                                         ;
                       55            01   D0 0004C        MOVL    #1, SIGN_FLAG                              ; 1252
          54           53            04   C7 0004F        DIVL3   #4, R3, R4                                 ; 1254
                       50            01   CE 00053        MNEGL   #1, I                                      ;
                                     05   11 00056        BRB     2$                                         ;
                     6740          6740   D2 00058 1$:    MCOML   (VALUE_PTR)[I], (VALUE_PTR)[I]             ; 1256
          F7           50            54   F2 0005D 2$:    AOBLSS  R4, I, 1$                                  ; 1255
          54           53            03   78 00061 3$:    ASHL    #3, R3, R4                                 ; 1258
                       54            02   C2 00065        SUBL2   #2, R4                                     ;
                       52            01   CE 00068        MNEGL   #1, I                                      ;
                                     32   11 0006B        BRB     6$                                         ;
 50       67           01            52   EF 0006D 4$:    EXTZV   I, #1, (VALUE_PTR), R0                     ; 1261
                       01            50   D1 00072        CMPL    R0, #1                                     ;
                                     14   12 00075        BNEQ    5$                                         ;
                               0240  8F   BB 00077        PUSHR   #^M<R6,R9>                                 ; 1266
                 FEA5  CF            02   FB 0007B        CALLS   #2, ADDC                                   ;
                       58            50   D0 00080        MOVL    R0, NEW_STRING                             ;
                                     56   DD 00083        PUSHL   STRING                                     ; 1267
                       6B            01   FB 00085        CALLS   #1, DBG$REL_MEMORY                         ;
                       56            58   D0 00088        MOVL    NEW_STRING, STRING                         ; 1268
                                     59   DD 0008B 5$:    PUSHL   POWER_OF_TWO                               ; 1271
                                     59   DD 0008D        PUSHL   POWER_OF_TWO                               ;
                 FE91  CF            02   FB 0008F        CALLS   #2, ADDC                                   ;
                       5A            50   D0 00094        MOVL    R0, NEW_POWER_OF_TWO                       ;
                                     59   DD 00097        PUSHL   POWER_OF_TWO                               ; 1272
                       6B            01   FB 00099        CALLS   #1, DBG$REL_MEMORY                         ;
                       59            5A   D0 0009C        MOVL    NEW_POWER_OF_TWO, POWER_OF_TWO             ; 1273
```

```
                    CA          52             54  F3 0009F 6$:     AOBLEQ  R4, I, 4$                                   ; 1258
                                01             55  D1 000A3         CMPL    SIGN_FLAG, #1                               ; 1275
                                               4D  12 000A6         BNEQ    9$                                          ;
                                53             04  C6 000A8         DIVL2   #4, R3                                      ; 1280
                                50             01  CE 000AB         MNEGL   #1, I                                       ;
                                               05  11 000AE         BRB     8$                                          ;
                              6740           6740  D2 000B0 7$:     MCOML   (VALUE_PTR)[I], (VALUE_PTR)[I]              ; 1282
                    F7          50             53  F2 000B5 8$:     AOBLSS  R3, I, 7$                                   ; 1281
                                    00000000'  EF  9F 000B9         PUSHAB  P.AAM                                       ; 1285
                                               56  DD 000BF         PUSHL   STRING                                      ; 1284
                          FE5F  CF             02  FB 000C1         CALLS   #2, ADDC                                    ;
                                58             50  D0 000C6         MOVL    R0, NEW_STRING                              ;
                                50             68  9A 000C9         MOVZBL  (NEW_STRING), R0                            ; 1287
                                50             02  C0 000CC         ADDL2   #2, R0                                      ;
                                50             04  C6 000CF         DIVL2   #4, R0                                      ;
                                           01  A0  9F 000D2         PUSHAB  1(R0)                                       ;
                      00000000G 00             01  FB 000D5         CALLS   #1, DBG$GET_MEMORY                          ;
                                56             50  D0 000DC         MOVL    R0, STRING                                  ;
                    66          68             01  81 000DF         ADDB3   #1, (NEW_STRING), (STRING)                  ; 1288
                            01  A6             2D  90 000E3         MOVB    #45, 1(STRING)                              ; 1289
                                50             68  9A 000E7         MOVZBL  (NEW_STRING), R0                            ; 1290
               02   A6      01  A8             50  28 000EA         MOVC3   R0, 1(NEW_STRING), 2(STRING)                ;
                                               58  DD 000F0         PUSHL   NEW_STRING                                  ; 1291
                                6B             01  FB 000F2         CALLS   #1, DBG$REL_MEMORY                          ;
                                               59  DD 000F5 9$:     PUSHL   POWER_OF_TWO                                ; 1293
                                6B             01  FB 000F7         CALLS   #1, DBG$REL_MEMORY                          ;
                                50             56  D0 000FA         MOVL    STRING, R0                                  ; 1295
                                                   04 000FD         RET                                                 ;

; Routine Size:  254 bytes,    Routine Base:  DBG$CODE + 0654
```

```
  1 ! THE REGISTER TABLE HOLDS ONE ENTRY PER REGISTER. EACH ENTRY IS MADE
  1 ! UP OF ONE LONGWORD. THE FIRST BYTE HOLDS THE CHARACTER COUNT OF
  1 ! THE REGISTER NAME. THE SECOND THROUGH FOURTH BYTES HOLD THE REGISTER
  1 ! NAME STRING. A SAMPLE ENTRY FOLLOWS:
  1 !
  1 !          -------------------------------------------------
  1 !          !              !     0     !     R     !     2     !
  1 !          !-------------------------------------------------!
  1 !--
  1
  1 MACRO
M 1          register_entry (string) =
  1                  %CHARCOUNT (STRING), %ASCII STRING, REP 3 - %CHARCOUNT (STRING) OF BYTE (0)%;
  1
  1 BIND
  1          register_table = UPLIT BYTE (
  1
  1                  register_entry ('R0'),
  1                  register_entry ('R1'),
  1                  register_entry ('R2'),
  1                  register_entry ('R3'),
  1                  register_entry ('R4'),
  1                  register_entry ('R5'),
  1                  register_entry ('R6'),
  1                  register_entry ('R7'),
  1                  register_entry ('R8'),
  1                  register_entry ('R9'),
  1                  register_entry ('R10'),
  1                  register_entry ('R11'),
  1                  register_entry ('AP'),
  1                  register_entry ('FP'),
  1                  register_entry ('SP'),
  1                  register_entry ('PC'),
  1                  register_entry ('PSL')):
  1
  1          BLOCK [, LONG];
  1
  1 !++
  1 ! THESE FIELD DEFINITIONS CONTROL ACCESS TO THE REGISTER TABLE
  1 !--
  1 MACRO
  1          REG_NAME        =8, 24, 0%,
  1          ctd_reg_name    =0, 24, 0%,
  1          REG_CH_CNT      =0, 8, 0%;
  1
  1 !++
  1 ! COMMON ASCII COUNTED STRINGS USED IN FAO CALLS.
  1 !--
  1
  1 BIND
  1          cs_ascii        = UPLIT ( %ASCIC '!AD'),
  1          colon_tab_stg   = UPLIT ( %ASCIC ':    ');
  1
  1 !++
  1 ! The following macros translate addresses to register offsets and
  1 ! vice versa.
  1 !--
```

```
MACRO
        this_is_reg (location) =
            (((location) GEQA dbg$runframe [dbg$l_user_r0]) AND
                ((location) LEQA dbg$runframe [dbg$l_user_psl]))%,

        reg_offset (location) =
            (location - dbg$runframe [dbg$l_user_regs]) / %UPVAL%;
```

```
1 GLOBAL ROUTINE dbg$out_regname( address ) =
1 !++
1 ! Functional Description
1 !
1 !       Given an address, see if it falls within the current
1 !  runframe in such a way as one could say that this address
1 !  corresponds to one of the general registers.  If this is
1 !  not the case, we return FALSE.  Otherwise we output the
1 !  name of the indicated register and return TRUE.
1 !
1 ! Formal Parameters:
1 !
1 !       address -the address which we are trying to symbolize
1 ! Implicit Inputs:
1 !
1 !       The format and use of the register table
1 !  which is local to this module.
1 !
1 ! Return Value
1 !       TRUE or FALSE.  See above.
1 !
1 ! Side Effects:
1 !
1 !       We may output a register name to the output buffer.
1 !
1 !--
2 BEGIN
2       BIND
2               register_vector = dbg$runframe [ DBG$L_USER_REGS ] : VECTOR;
2       LOCAL
2               reg_index;
2
3       IF( NOT this_is_reg(.address) )
2       then
2               return(false);
2
2       reg_index = reg_offset(.address);
2
2       ! Check that the address EXACTLY matches one which
2       ! we currently bind to a register name.
2
3       IF( register_vector[.reg_index] NEQA .address )
2       then
2               return(FALSE);
2
2       ! An exact match has been found.
2       ! Output the register name and return
2       ! a success status.
2
2       dbg$print( UPLIT( %ASCIC '!AC' ),
2               register_table[.reg_index, ctd_reg_name ]);
2
2       return(TRUE);
1 END;
```

```
                                              .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                   02  0009E P.AAN:   .BYTE   2
               30  52  0009F          .ASCII  \R0\
                   00  000A1          .BYTE   0
                   02  000A2          .BYTE   2
               31  52  000A3          .ASCII  \R1\
                   00  000A5          .BYTE   0
                   02  000A6          .BYTE   2
               32  52  000A7          .ASCII  \R2\
                   00  000A9          .BYTE   0
                   02  000AA          .BYTE   2
               33  52  000AB          .ASCII  \R3\
                   00  000AD          .BYTE   0
                   02  000AE          .BYTE   2
               34  52  000AF          .ASCII  \R4\
                   00  000B1          .BYTE   0
                   02  000B2          .BYTE   2
               35  52  000B3          .ASCII  \R5\
                   00  000B5          .BYTE   0
                   02  000B6          .BYTE   2
               36  52  000B7          .ASCII  \R6\
                   00  000B9          .BYTE   0
                   02  000BA          .BYTE   2
               37  52  000BB          .ASCII  \R7\
                   00  000BD          .BYTE   0
                   02  000BE          .BYTE   2
               38  52  000BF          .ASCII  \R8\
                   00  000C1          .BYTE   0
                   02  000C2          .BYTE   2
               39  52  000C3          .ASCII  \R9\
                   00  000C5          .BYTE   0
                   03  000C6          .BYTE   3
           30  31  52  000C7          .ASCII  \R10\
                   03  000CA          .BYTE   3
           31  31  52  000CB          .ASCII  \R11\
                   02  000CE          .BYTE   2
               50  41  000CF          .ASCII  \AP\
                   00  000D1          .BYTE   0
                   02  000D2          .BYTE   2
               50  46  000D3          .ASCII  \FP\
                   00  000D5          .BYTE   0
                   02  000D6          .BYTE   2
               50  53  000D7          .ASCII  \SP\
                   00  000D9          .BYTE   0
                   02  000DA          .BYTE   2
               43  50  000DB          .ASCII  \PC\
                   00  000DD          .BYTE   0
                   03  000DE          .BYTE   3
           4C  53  50  000DF          .ASCII  \PSL\
                       000E2          .BLKB   2
       44  41  21  03  000E4 P.AAO:   .ASCII  <3>\!AD\
       20  20  3A  03  000E8 P.AAP:   .ASCII  <3>\:  \
       43  41  21  03  000EC P.AAQ:   .ASCII  <3>\!AC\

                             REGISTER_TABLE=      P.AAN
                             CS_ASCII=            P.AAO
```

```
                                                             COLON_TAB_STG=      P.AAP

                                                                    .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                                  0004 00000        .ENTRY  DBG$OUT_REGNAME, Save R2                                  ; 1360
                                     52 00000000G  00  9E 00002     MOVAB   DBG$RUNFRAME+4, R2                                        ;
                                     50            62  9E 00009     MOVAB   DBG$RUNFRAME+4, R0                                        ; 1392
                                     50         04 AC  D1 0000C     CMPL    ADDRESS, R0                                               ;
                                                   37  1F 00010     BLSSU   1$                                                        ;
                                     50         40 A2  9E 00012     MOVAB   DBG$RUNFRAME+68, R0                                       ;
                                     50         04 AC  D1 00016     CMPL    ADDRESS, R0                                               ;
                                                   2D  1A 0001A     BGTRU   1$                                                        ;
                                     50            62  9E 0001C     MOVAB   DBG$RUNFRAME+4, R0                                        ; 1396
                        50        04 AC            50  C3 0001F     SUBL3   R0, ADDRESS, R0                                           ;
                                     50            04  C6 00024     DIVL2   #4, REG_INDEX                                             ;
                                     51          6240  DE 00027     MOVAL   REGISTER_VECTOR[REG_INDEX], R1                            ; 1401
                                  04 AC            51  D1 0002B     CMPL    R1, ADDRESS                                               ;
                                                   18  12 0002F     BNEQ    1$                                                        ;
                                        00000000'EF40  DF 00031     PUSHAL  REGISTER_TABLE[REG_INDEX]                                 ; 1410
                                        00000000'  EF  9F 00038     PUSHAB  P.AAQ                                                     ; 1409
                          00000000G  00            02  FB 0003E     CALLS   #2, DBG$PRINT                                             ; 1410
                                     50            01  D0 00045     MOVL    #1, R0                                                    ; 1412
                                                       04 00048     RET                                                               ;
                                                   50  D4 00049 1$: CLRL    R0                                                        ; 1413
                                                       04 0004B     RET                                                               ;

; Routine Size:  76 bytes,    Routine Base:  DBG$CODE + 0752
; 1294          1414    1 GLOBAL ROUTINE dbg$reg_match (string_desc) =
; 1295          1415    1 !++
; 1296          1416    1 ! Functional description:
; 1297          1417    1 !       Compares a string described by the string descriptor passed
; 1298          1418    1 !       as the routine formal to each of the names of the machine
; 1299          1419    1 !       registers. If the string matches a register name, return the
; 1300          1420    1 !       number of the register (0-16, where 16 is the PSL). Otherwise,
; 1301          1421    1 !       return a -1.
; 1302          1422    1 !
; 1303          1423    1 ! Inputs:
; 1304          1424    1 !       string_desc     - string descriptor to symbol string
; 1305          1425    1 !
; 1306          1426    1 ! Implicit inputs:
; 1307          1427    1 !       the VAX machine register table
; 1308          1428    1 !
; 1309          1429    1 ! Implicit outputs:
; 1310          1430    1 !       none
; 1311          1431    1 !
; 1312          1432    1 ! Routine value:
; 1313          1433    1 !       The register number 0 - 16 if a match is found.
; 1314          1434    1 !       -1 if no match is found.
; 1315          1435    1 !
; 1316          1436    1 ! Side effects:
; 1317          1437    1 !       none
; 1318          1438    1 !--
; 1319          1439    1
; 1320          1440    2         BEGIN
; 1321          1441    2
; 1322          1442    2         MAP
; 1323          1443    2                 string_desc : REF BLOCK [, BYTE];
; 1324          1444    2
; 1325          1445    2
; 1326          1446    2
; 1327          1447    2         INCR index from 0 to register_count-1 DO
; 1328          1448    3             BEGIN
; 1329          1449    3             IF ch$eql (.string_desc[dsc$w_length], ch$ptr(.string_desc [dsc$a_pointer]),
; 1330          1450    3                         .register_table[.index, reg_ch_cnt],
; 1331          1451    3                         ch$ptr(register_table[.index, reg_name]))
; 1332          1452    3             THEN
; 1333          1453    3                 RETURN .index;
; 1334          1454    2             END;
; 1335          1455    2
; 1336          1456    2         RETURN (-1);
; 1337          1457    2
; 1338          1458    1         END;



                              003C 00000          .ENTRY  DBG$REG_MATCH, Save R2,R3,R4,R5              ; 1414
                   55      04   AC   D0 00002          MOVL    STRING_DESC, R5                              ; 1449
                                54   D4 00006          CLRL    INDEX                                        ;
                00000000'EF44   DF 00008 1$:      PUSHAL  REGISTER_TABLE[INDEX]                        ; 1450
                   50           9E   9A 0000F          MOVZBL  @(SP)+, R0                                   ;
                00000000'EF44   DF 00012          PUSHAL  REGISTER_TABLE+1[INDEX]                      ; 1451
      50          00       04   B5   04   BC   2D 00019          CMPC5   @STRING_DESC, @4(R5), #0, R0, @(SP)+       ;
```

```
                                        9E      00020                                              ;
                                        04   12 00021          BNEQ    2$                          ;
                              50        54   D0 00023          MOVL    INDEX, R0                   ; 1453
                                             04 00026          RET                                 ;
                   DD         54        10   F3 00027 2$:      AOBLEQ  #16, INDEX, 1$              ; 1447
                              50        01   CE 0002B          MNEGL   #1, R0                      ; 1456
                                             04 0002E          RET                                 ; 1458

; Routine Size:  47 bytes,    Routine Base:  DBG$CODE + 079E
```

```
GLOBAL ROUTINE dbg$output_psl (value) : NOVALUE =
!++
! FUNCTIONAL DESCRIPTION:
!       Formats and outputs two lines of specially
!       formatted data contained in the PSL. The fields shown are
!
!               CMP     - compatibility mode
!               TP      - trace trap pending
!               FPD     - first part done
!               IS      - interrupt stack
!               CURMOD  - current access mode
!               PRVMOD  - previous access mode
!               IPL     - interrupt priority level
!               DV      - decimal overflow trap enable
!               FU      - floating underflow trap enable
!               IV      - integer overflow trap enable
!               T       - trace trap
!               N       - condition code
!               Z       - condition code
!               V       - condition code
!               C       - condition code
!
! CALLING SEQUENCE:
!       dbg$output_psl ()
!
! INPUTS:
!       value   -The current contents of the PSL
!
! IMPLICIT INPUTS:
!       NONE
!
! OUTPUTS:
!       NONE
!
! IMPLICIT OUTPUTS:
!       NONE
!
! ROUTINE value:
!       NONE
!       NOVALUE
!
! SIDE EFFECTS:
!       Two lines are output to the output device.
!--

BEGIN
        MAP
                value: BLOCK;
        MACRO
                position_field  = 0, 8, 1%,
                size_field      = 8, 8, 1%,
                mode_field      = 16, 4, 1%,
                blanks_field    = 20, 4, 1%,
                width_field     = 24, 8, 1%,

                psl_field (name, position, size, mode, width, leading_blanks) =
                        position, size, mode OR (leading_blanks ^ 4), width%;
```

```
LITERAL
        decimal         = 0,
        max_psl_field   = 15;

BIND
        psl_table = UPLIT BYTE (

        psl_field (CMP, 31, 1, 0, 1, 1),
        psl_field (TP, 30, 1, 0, 1, 3),
        psl_field (FPD, 27, 1, 0, 1, 2),
        psl_field (IS, 26, 1, 0, 1, 3),
        psl_field (CURMOD, 24, 2, 1, 4, 2),
        psl_field (PRVMOD, 22, 2, 1, 4, 3),
        psl_field (IPL, 16, 5, 0, 2, 3),
        psl_field (DV, 7, 1, 0, 1, 2),
        psl_field (FU, 6, 1, 0, 1, 2),
        psl_field (IV, 5, 1, 0, 1, 2),
        psl_field (T, 4, 1, 0, 1, 1),
        psl_field (N, 3, 1, 0, 1, 1),
        psl_field (Z, 2, 1, 0, 1, 1),
        psl_field (V, 1, 1, 0, 1, 1),
        psl_field (C, 0, 1, 0, 1, 1))

        : BLOCK,

        hex_number      = UPLIT BYTE (%ASCIC '!AD!#XB'),
        stg_desc        = UPLIT BYTE (%ASCIC '!AD!AD'),
        blanks          = UPLIT BYTE (%ASCII '     '),

        priv_modes      = UPLIT BYTE (

        %ASCII 'KERN',
        %ASCII 'EXEC',
        %ASCII 'SUPR',
        %ASCII 'USER')

        : VECTOR;

! Write out the standard title which describes the PSL fields.

dbg$print( UPLIT (%ASCIC '!_CMP TP FPD IS CURMOD PRVMOD IPL DV FU IV T N Z V C'));
dbg$newline();
dbg$print (UPLIT (%ASCIC '! '));
INCR count FROM 0 TO max_psl_field - 1 DO
        IF .psl_table [.count, mode_field] EQL decimal
        THEN
                BEGIN
                dbg$print (hex_number,
                        .psl_table [.count, blanks_field], blanks,
                        .psl_table [.count, width_field],
                        .value [0, .psl_table [.count, position_field],
                        .psl_table [.count, size_field], 0]);
                END
        ELSE
                BEGIN
                dbg$print (stg_desc,
                        .psl_table [.count, blanks_field], blanks,
; 1454            1573  3                                                  .psl_table [.count, width_field],
; 1455            1574  3                                                  priv_modes [.value [0, .psl_table [.count, position_field],
; 1456            1575  3                                                          .psl_table [.count, size_field], 0]]);
; 1457            1576  2                                          END;
; 1458            1577  1 END;



                                                                                     .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

30  01  1A  01  20  01  1B  01  30  01  1E  01  10  01  1F  000F0 P.AAR:   .BYTE   31, 1, 16, 1, 30, 1, 48, 1, 27, 1, 32, 1, - ;
01  07  02  30  05  10  04  31  02  16  04  21  02  18  01  000FF                  26, 1, 48, 1, 24, 2, 33, 4, 22, 2, 49, 4, - ;
03  01  10  01  04  01  20  01  05  01  20  01  06  01  20  0010E                  16, 5, 48, 2, 7, 1, 32, 1, 6, 1, 32, 1, -  ;
01  10  01  00  01  10  01  01  01  10  01  02  01  10  01  0011D                  5, 1, 32, 1, 4, 1, 16, 1, 3, 1, 16, 1, 2, - ;
                                                                                   1, 16, 1, 1, 1, 16, 1, 0, 1, 16, 1           ;
                                42  58  23  21  44  41  21  07  0012C P.AAS:   .ASCII  <7>\!AD!#XB\                                 ;
                                    44  41  21  44  41  21  06  00134 P.AAT:   .ASCII  <6>\!AD!AD\                                  ;
                                                20  20  20  20  0013B P.AAU:   .ASCII  \    \                                       ;
                                                4E  52  45  4B  0013F P.AAV:   .ASCII  \KERN\                                       ;
                                                43  45  58  45  00143          .ASCII  \EXEC\                                       ;
                                                52  50  55  53  00147          .ASCII  \SUPR\                                       ;
                                                52  45  53  55  0014B          .ASCII  \USER\                                       ;
                                                                0014F          .BLKB   1                                            ;
49  20  44  50  46  20  50  54  20  50  4D  43  5F  21  34  00150 P.AAW:   .ASCII  \4!_CMP TP FPD IS CURMOD PRVMOD IPL DV FU\   ;
44  4F  4D  56  52  50  20  44  4F  4D  52  55  43  20  53  0015F                                                               ;
                        55  46  20  56  44  20  4C  50  49  20  0016E                                                               ;
00  00  43  20  56  20  5A  20  4E  20  54  20  56  49  20  00178          .ASCII  \ IV T N Z V C\<0><0><0>                     ;
                                                            00  00187                                                               ;
                                                00  5F  21  02  00188 P.AAX:   .ASCII  <2>\!_\<0>                                   ;

                                                                  PSL_TABLE=          P.AAR
                                                                  HEX_NUMBER=         P.AAS
                                                                  STG_DESC=           P.AAT
                                                                  BLANKS=             P.AAU
                                                                  PRIV_MODES=         P.AAV


                                                                                     .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                                   003C 00000          .ENTRY  DBG$OUTPUT_PSL, Save R2,R3,R4,R5               ; 1459
                            55 00000000G      00    9E 00002          MOVAB   DBG$PRINT, R5                                   ;
                            54 00000000'      EF    9E 00009          MOVAB   BLANKS, R4                                      ;
                                        15    A4    9F 00010          PUSHAB  P.AAW                                           ; 1556
                            65                01    FB 00013          CALLS   #1, DBG$PRINT                                   ;
                   00000000G 00               00    FB 00016          CALLS   #0, DBG$NEWLINE                                 ; 1557
                                        4D    A4    9F 0001D          PUSHAB  P.AAX                                           ; 1558
                            65                01    FB 00020          CALLS   #1, DBG$PRINT                                   ;
                                              52    D4 00023          CLRL    COUNT                                           ; 1559
                            53          B5 A442     DE 00025 1$:      MOVAL   PSL_TABLE[COUNT], R3                            ; 1560
                            51                63    98 0002A          CVTBL   (R3), R1                                        ; 1567
     50      04   AC      01  A3              51    EF 0002D          EXTZV   R1, 1(R3), VALUE, R0                            ;
                            0F          02    A3    93 00034          BITB    2(R3), #15                                      ; 1560
                                              12    12 00038          BNEQ    2$                                              ;
                                              50    DD 0003A          PUSHL   R0                                              ; 1566
                            7E          03    A3    98 0003C          CVTBL   3(R3), -(SP)                                    ; 1565
                                              54    DD 00040          PUSHL   R4                                              ; 1563
     7E           63          04              14    EE 00042          EXTV    #20, #4, (R3), -(SP)                            ; 1564
```

```
                                                   F1    A4   9F 00047          PUSHAB  HEX_NUMBER                          ; 1563
                                                         12   11 0004A          BRB     3$                                  ;
                                                   04 A440    DF 0004C  2$:     PUSHAL  PRIV_MODES[R0]                      ; 1575
                                         7E        03    A3   9B 00050          CVTBL   3(R3), -(SP)                        ;
                                                         54   DD 00054          PUSHL   R4                                  ; 1571
          7E               63            04              14   EE 00056          EXTV    #20, #4, (R3), -(SP)                ; 1575
                                                   F9    A4   9F 0005B          PUSHAB  STG_DESC                            ; 1571
                                         65              05   FB 0005E  3$:     CALLS   #5, DBG$PRINT                       ; 1575
                           C0            52              0E   F3 00061          AOBLEQ  #14, COUNT, 1$                      ; 1560
                                                              04 00065          RET                                         ; 1577

; Routine Size:  102 bytes,    Routine Base:  DBG$CODE + 07CD
```

```
1 GLOBAL ROUTINE dbg$digit_scan(a,l,nd)=
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine will scan the string pointed to by 'a' with
1 !       length 'l' to determine if it is a valid digit string.
1 !       it will also build a new descriptor for the input
1 !
1 ! INPUTS:
1 !
1 !       a - address of string
1 !       l - length of string
1 !       nd - pointer to new descriptor
1 !
1 ! IMPLICIT INPUTS:
1 !
1 ! OUTPUTS:
1 !       none
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 ! ROUTINE VALUE:
1 !       1 - a valid digit string
1 !       3 - a valid digit string beginning with a sign   + or -
1 !       0 - not a valid digit string
1 !
1 ! SIDE EFFECTS:
1 !
1 !--
2 BEGIN
2 builtin cvtsp,cvtpl;
2 MAP  a : ref vector[,byte],nd : ref block[,byte];
2 LOCAL s,i,dp,dpp,ep,esn,p : vector [40,byte],ln,f,nwl;
2 BIND max_packed_size = uplit(31);
2
2 ! Quick fix for a problem: this routine was returning "true" on the
2 ! string "e".
2 !
3 IF .L EQL 1 AND (.A[0] EQL 'E' OR .A[0] EQL 'e')
2 THEN
2     RETURN 0;
2
2     i = 0 ;
2     s = 1 ;
2     f = dp=ep=dpp=esn=0;
2 !
2 ! get possible trailing spaces
2     nwl = .l ;
2     WHILE .a[.nwl-1] EQL %c' '
2     DO
2         if (nwl = .nwl-1) leq 0 then return 0;
2 ! skip over possible leading spaces
2     WHILE .a[.i] eql %c' '
2         DO
3         BEGIN
3         INCR c from 0 to .nwl-1
3             DO
```

```
3                 a[.c] = .a[.c+1];
3             if (nwl = .nwl-1) leq 0 then return 0;
2             END;
2
2         nd[dsc$w_length] = .nd[dsc$w_length] - (.l-.nwl);
2
2         IF .a[.i] eql %C'+' OR .a[.i] eql %C'-'
2         THEN
3             BEGIN
3             s = 3 ;
3             i = .i + 1 ;
3             nd[dsc$b_dtype] = dsc$k_dtype_nl ;
3             END
2         ELSE
2             nd[dsc$b_dtype] = dsc$k_dtype_nro ;
2
2         INCR c from .i to .nwl-1
2         DO
3             BEGIN
3             if .a[.c] eql %c'.'
3             then
3                 if .dp
3                 then
3                     return 0
3                 else
4                     begin
4                     dp = 1;
4                     dpp = .c;
4                     end
3             else
3                 if .a[.c] eql %c'd' or .a[.c] eql %c'D' or
3                     .a[.c] eql %c'e' or .a[.c] eql %c'E'
3                 then
3                     if .ep neq 0
3                     then
3                         return 0
3                     else
3                         ep = .c
3                 else
3                     if .a[.c] eql %c'+' or .a[.c] eql %c'-'
3                     then
3                         if .esn neq 0 or .ep eql 0
3                         then
3                             return 0
3                         else
4                             esn =(if .a[.c] eql %c'+' then 1 else -1)
3                     else
4                         if not (.a[.c] geq %c'0' and .a[.c] leq %c'9')
3                         then
3                             return 0
2             END;
2
2   ! now construct scale factor for desc and redo the length
2
2
2         if .ep neq 0
2         then
```

```
; 1574          1692 3          begin
; 1575          1693 3          i =.nwl-.ep-1;
; 1576          1694 3          if .esn eql 0
; 1577          1695 3          then
; 1578          1696 4              begin
; 1579          1697 4              a[.ep] = %c'+';
; 1580          1698 4              cvtsp(i,a[.ep],max_packed_size,p[0]);
; 1581          1699 4              end
; 1582          1700 3          else
; 1583          1701 4              begin
; 1584          1702 4              i = .i -1 ;
; 1585          1703 4              cvtsp(i,a[.ep+1],max_packed_size,p[0]);
; 1586          1704 3              end;
; 1587          1705 3          cvtpl(max_packed_size,p[0],ln);
; 1588          1706 3          nd[dsc$b_scale] = .ln ;
; 1589          1707 3          nd[dsc$w_length] = .ep ;
; 1590          1708 3          nwl = .ep ;
; 1591          1709 2          end;
; 1592          1710 2
; 1593          1711 2
; 1594          1712 2      if .dp eql 0
; 1595          1713 2      then
; 1596          1714 2          0
; 1597          1715 2      else
; 1598          1716 3          begin
; 1599          1717 3          ln = (.nwl-.dpp-1);
; 1600          1718 3          nd[dsc$b_scale] = .nd[dsc$b_scale] - .ln ;
; 1601          1719 3          nd[dsc$w_length] = .nd[dsc$w_length] - 1 ;
; 1602          1720 3          ch$move(.ln,a[.dpp+1],p[0]);
; 1603          1721 3          ch$move(.ln,p[0],a[.dpp]);
; 1604          1722 2          end;
; 1605          1723 2      if .nd[dsc$b_dtype] eql dsc$k_dtype_nl
; 1606          1724 2      then
; 1607          1725 2          nd[dsc$w_length] = .nd[dsc$w_length] - 1;
; 1608          1726 2
; 1609          1727 2
; 1610          1728 2      return .s                               ! catch all return
; 1611          1729 1 END;             ! End of digit_scan



                                                           .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                                   0000001F  0018C P.AAY:  .LONG   31                                        ;

                                                     MAX_PACKED_SIZE=    P.AAY


                                                           .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                       0FFC 00000          .ENTRY  DBG$DIGIT_SCAN, Save R2,R3,R4,R5,R6,R7,R8,- ; 1578
                                                                   R9,R10,R11                                :
                  5E              2C   C2 00002            SUBL2   #44, SP                                   :
                  01        08    AC   D1 00005            CMPL    L, #1                                     ; 1616
                                  0E   12 00009            BNEQ    1$                                        :
            45    8F        04    BC   91 0000B            CMPB    @A, #69                                   :
                                  3E   13 00010            BEQL    6$                                        :
```

```
                 65   8F    04    BC  91 00012       CMPB    @A, #101                        ;
                                  37  13 00017       BEQL    6$                              ;
                      6E          01  D0 00019 1$:   MOVL    #1, S                           ; 1621
                                  53  7C 0001C       CLRQ    ESN                             ; 1622
                                  59  7C 0001E       CLRQ    EP                              ;
                                  5B  D4 00020       CLRL    DP                              ;
                                  50  D4 00022       CLRL    F                               ;
                      55    08    AC  D0 00024       MOVL    L, NWL                          ; 1625
                      57    04    AC  D0 00028       MOVL    A, R7                           ; 1626
                      20    FF  A547  91 0002C 2$:   CMPB    -1(NWL)[R7], #32                ;
                                  05  12 00031       BNEQ    3$                              ;
                      F6          55  F5 00033       SOBGTR  NWL, 2$                         ; 1628
                                  18  11 00036       BRB     6$                              ;
                      20        6447  91 00038 3$:   CMPB    (I)[R7], #32                    ; 1630
                                  15  12 0003C       BNEQ    7$                              ;
                      50          01  CE 0003E       MNEGL   #1, C                           ; 1633
                                  06  11 00041       BRB     5$                              ;
                    6047    01  A047  90 00043 4$:   MOVB    1(C)[R7], (C)[R7]               ; 1635
        F6            50          55  F2 00049 5$:   AOBLSS  NWL, C, 4$                      ;
                      E8          55  F5 0004D       SOBGTR  NWL, 3$                         ; 1636
                                0101  31 00050 6$:   BRW     25$                             ;
                      58    0C    AC  D0 00053 7$:   MOVL    ND, R8                          ; 1639
        50            55    08    AC  C3 00057       SUBL3   L, NWL, R0                      ;
                      68          50  A0 0005C       ADDW2   R0, (R8)                        ;
                      2B        6447  91 0005F       CMPB    (I)[R7], #43                    ; 1641
                                  06  13 00063       BEQL    8$                              ;
                      2D        6447  91 00065       CMPB    (I)[R7], #45                    ;
                                  0B  12 00069       BNEQ    9$                              ;
                      6E          03  D0 0006B 8$:   MOVL    #3, S                           ; 1644
                                  54  D6 0006E       INCL    I                               ; 1645
                 02   A8          10  90 00070       MOVB    #16, 2(R8)                      ; 1646
                                  04  11 00074       BRB     10$                             ; 1641
                 02   A8          13  90 00076 9$:   MOVB    #19, 2(R8)                      ; 1649
                      50    FF    A4  9E 0007A 10$:  MOVAB   -1(R4), C                       ; 1651
                                  64  11 0007E       BRB     19$                             ;
                      51        6047  9A 00080 11$:  MOVZBL  (C)[R7], R1                     ; 1654
                      2E          51  91 00084       CMPB    R1, #46                         ;
                                  0B  12 00087       BNEQ    12$                             ;
                      C4          5B  E8 00089       BLBS    DP, 6$                          ; 1656
                      5B          01  D0 0008C       MOVL    #1, DP                          ; 1661
                      5A          50  D0 0008F       MOVL    C, DPP                          ; 1662
                                  50  11 00092       BRB     19$                             ; 1656
                 64   8F          51  91 00094 12$:  CMPB    R1, #100                        ; 1665
                                  12  13 00098       BEQL    13$                             ;
                 44   8F          51  91 0009A       CMPB    R1, #68                         ;
                                  0C  13 0009E       BEQL    13$                             ;
                 65   8F          51  91 000A0       CMPB    R1, #101                        ; 1666
                                  06  13 000A4       BEQL    13$                             ;
                 45   8F          51  91 000A6       CMPB    R1, #69                         ;
                                  09  12 000AA       BNEQ    14$                             ;
                                  59  D5 000AC 13$:  TSTL    EP                              ; 1668
                                  A0  12 000AE       BNEQ    6$                              ;
                      59          50  D0 000B0       MOVL    C, EP                           ; 1672
                                  2F  11 000B3       BRB     19$                             ; 1668
                                  52  D4 000B5 14$:  CLRL    R2                              ; 1674
                      2B          51  91 000B7       CMPB    R1, #43                         ;
                                  04  12 000BA       BNEQ    15$                             ;
```

```
                                                                      N  4
DBGLEVEL1                                                             16-Sep-1984 01:27:02    VAX-11 Bliss-32 V4.0-742                  Page  55
V04-000                                                               14-Sep-1984 12:17:02    [DEBUG.SRC]DBGLEVEL1.B32;1                    (16)

                                                          52  D6 000BC        INCL    R2                                      ;
                                                          05  11 000BE        BRB     16$                                     ;
                                            2D            51  91 000C0 15$:   CMPB    R1, #45                                 ;
                                                          15  12 000C3        BNEQ    18$                                     ;
                                                          53  D5 000C5 16$:   TSTL    ESN                                     ; 1676
                                                          87  12 000C7        BNEQ    6$                                      ;
                                                          59  D5 000C9        TSTL    EP                                      ;
                                                          83  13 000CB        BEQL    6$                                      ;
                                            05            52  E9 000CD        BLBC    R2, 17$                                 ; 1680
                                            53            01  D0 000D0        MOVL    #1, ESN                                 ;
                                                          0F  11 000D3        BRB     19$                                     ;
                                            53            01  CE 000D5 17$:   MNEGL   #1, ESN                                 ;
                                                          0A  11 000D8        BRB     19$                                     ; 1676
                                            30            51  91 000DA 18$:   CMPB    R1, #48                                 ; 1682
                                                          75  1F 000DD        BLSSU   25$                                     ;
                                            39            51  91 000DF        CMPB    R1, #57                                 ;
                                                          70  1A 000E2        BGTRU   25$                                     ;
                              98            50            55  F2 000E4 19$:   AOBLSS  NWL, C, 11$                             ; 1653
                                                          59  D5 000E8        TSTL    EP                                      ; 1690
                                                          3E  13 000EA        BEQL    22$                                     ;
                              56            55            59  C3 000EC        SUBL3   EP, NWL, R6                             ; 1693
                                            54       FF   A6  9E 000F0        MOVAB   -1(R6), I                               ;
                                                          53  D5 000F4        TSTL    ESN                                     ; 1694
                                                          11  12 000F6        BNEQ    20$                                     ;
                                          6947            2B  90 000F8        MOVB    #43, (EP)[R7]                           ; 1697
        04   AE 00000000'     EF          6947            54  09 000FC        CVTSP   I, (EP)[R7], MAX_PACKED_SIZE, P         ; 1698
                                                          0E  11 00107        BRB     21$                                     ; 1694
                                                          54  D7 00109 20$:   DECL    I                                       ; 1702
        04   AE 00000000'     EF       01 A947            54  09 0010B        CVTSP   I, 1(EP)[R7], MAX_PACKED_SIZE, P        ; 1703
                              56       04   AE 00000000'  EF  36 00117 21$:   CVTPL   MAX_PACKED_SIZE, P, LN                  ; 1705
                                       08   A8            56  90 00120        MOVB    LN, 8(R8)                               ; 1706
                                            68            59  B0 00124        MOVW    EP, (R8)                                ; 1707
                                            55            59  D0 00127        MOVL    EP, NWL                                 ; 1708
                                                          5B  D5 0012A 22$:   TSTL    DP                                      ; 1712
                                                          1A  13 0012C        BEQL    23$                                     ;
                                            55            5A  C2 0012E        SUBL2   DPP, R5                                 ; 1717
                                            56       FF   A5  9E 00131        MOVAB   -1(R5), LN                              ;
                                       08   A8            56  82 00135        SUBB2   LN, 8(R8)                               ; 1718
                                                          68  B7 00139        DECW    (R8)                                    ; 1719
                         04   AE       01 AA47            56  28 0013B        MOVC3   LN, 1(DPP)[R7], P                       ; 1720
                            6A47       04   AE            56  28 00142        MOVC3   LN, P, (DPP)[R7]                        ; 1721
                                            10       02   A8  91 00148 23$:   CMPB    2(R8), #16                              ; 1723
                                                          02  12 0014C        BNEQ    24$                                     ;
                                                          68  B7 0014E        DECW    (R8)                                    ; 1725
                                            50            6E  D0 00150 24$:   MOVL    S, R0                                   ; 1728
                                                              04 00153        RET                                             ;
                                                          50  D4 00154 25$:   CLRL    R0                                      ; 1729
                                                              04 00156        RET                                             ;

; Routine Size:  343 bytes,    Routine Base:  DBG$CODE + 0833


; 1612          1730  1
```

```
1 GLOBAL ROUTINE DBG$MAP_TO_REG_ADDR (INPUT_ADDR, OUTPUT_ADDR) =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine examines the input address to see if it corresponds to some
1 !       address in the dbg$reg_values register save area maintained by the
1 !       routines dbg$sta_setcontext and dbg$sta_setregisters routines. If the
1 !       address represents some address in the register value save area, then
1 !       the address is mapped to the corresponding address in the dbg$l_user_regs
1 !       register save area.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       INPUT_ADDR      - A longword containing the address on which to attempt
1 !                         the mapping
1 !
1 !       OUTPUT_ADDR     - The address of a longword to contain the resulting mapped
1 !                         address
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       The address of the register context save area, dbg$reg_values, and the
1 !       address of the user runframe register save area, dbg$runframe [dbg$l_user_regs].
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !       An unsigned integer longword completion code
1 !
1 ! COMPLETION CODES:
1 !
1 !       STS$K_SUCCESS   (1)     - Success. Input address mapped, and mapped address
1 !                                 returned.
1 !
1 !       STS$K_SEVERE    (4)     - Failure. No mapping.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
2    BEGIN
2
2    LOCAL
2       RUNFRAME_ADDRS_VECT     : REF VECTOR [,BYTE]; ! Runframe regs area
2
2    runframe_addrs_vect = dbg$runframe [dbg$l_user_regs];
2
2    ! Check to see if the input address falls in the register context save area.
2
2    IF .input_addr GEQA dbg$reg_values [0]
2                       AND
2       .input_addr LSSA dbg$reg_values [17] ! The register vector has only 17 longwords
```

```
; 1671          1788  2     THEN
; 1672          1789  3         BEGIN
; 1673          1790  3
; 1674          1791  3         ! Input_addr definitely coresponds to some address in the context area.
; 1675          1792  3         ! Map it to the user runframe.
; 1676          1793  3
; 1677          1794  3         .output_addr = runframe_addrs_vect [.input_addr - dbg$reg_values [0]];
; 1678          1795  3
; 1679          1796  3         RETURN sts$k_success;
; 1680          1797  3         END
; 1681          1798  2     ELSE
; 1682          1799  3         BEGIN
; 1683          1800  3
; 1684          1801  3         ! No match
; 1685          1802  3
; 1686          1803  3         RETURN sts$k_severe;
; 1687          1804  2         END;
; 1688          1805  1     END;                ! End of dbg$map_to_reg_addr



                                          0004 00000         .ENTRY  DBG$MAP_TO_REG_ADDR, Save R2                 ; 1731
                      52 00000000G   00   9E 00002         MOVAB   DBG$REG_VALUES, R2
                      51 00000000G   00   9E 00009         MOVAB   DBG$RUNFRAME+4, RUNFRAME_ADDRS_VECT          ; 1781
                      50              62  9E 00010         MOVAB   DBG$REG_VALUES, R0                           ; 1785
                      50        04    AC  D1 00013         CMPL    INPUT_ADDR, R0                               ;
                                      1B  1F 00017         BLSSU   1$                                           ;
                      50        44    A2  9E 00019         MOVAB   DBG$REG_VALUES+68, R0                        ; 1787
                      50        04    AC  D1 0001D         CMPL    INPUT_ADDR, R0                               ;
                                      11  1E 00021         BGEQU   1$                                           ;
                      50              62  9E 00023         MOVAB   DBG$REG_VALUES, R0                           ; 1794
           50        04    AC         50  C3 00026         SUBL3   R0, INPUT_ADDR, R0                           ;
     08    BC              50         51  C1 0002B         ADDL3   RUNFRAME_ADDRS_VECT, R0, @OUTPUT_ADDR        ;
                      50              01  D0 00030         MOVL    #1, R0                                       ; 1799
                                          04 00033         RET                                                  ;
                      50              04  D0 00034 1$:     MOVL    #4, R0                                       ; 1803
                                          04 00037         RET                                                  ; 1805

; Routine Size:  56 bytes,    Routine Base:  DBG$CODE + 098A


; 1689          1806  1
```

```
GLOBAL ROUTINE DBG$EXACT_MAP_TO_REG (INPUT_ADDR, REG_ADDR) =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This routine checks to see if the input address can be mapped to the
!       exact starting address of one of the context register value save areas.
!       If it can, then the address is mapped to the starting address of the
!       corresponding runframe registers.
!
! FORMAL PARAMETERS:
!
!       INPUT_ADDR      - A longword containing the address to be mapped
!
!       REG_ADDR        - The address of a longword to contain the address of
!                         the mapped-to register
!
! IMPLICIT INPUTS:
!
!       dbg$runframe [dbg$l_user_regs] - the beginning address of the runframe
!                                        registers
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
!
!       An unsigned integer longword completion code
!
! COMPLETION CODES:
!
!       STS$K_SUCCESS   (1)     - Success. Input address mapped to register address.
!
!       STS$K_SEVERE    (4)     - Failure. Input address not mapped.
!
! SIDE EFFECTS:
!
!       NONE
!
!--
    BEGIN

    LOCAL
        RUNFRAME_ADDRESS;                                       ! Address within runframe
                                                                ! area

    ! See if the input address maps to any place in the runframe regs

    IF dbg$map_to_reg_addr (.input_addr, runframe_address)
    THEN
        BEGIN

        ! See if the resulting mapped address corresponds exactly to a reg
        ! beginning address

        IF ((.runframe_address - dbg$runframe [dbg$l_user_regs]) MOD 4) EQL 0
```

```
; 1748          1864  3             THEN
; 1749          1865  4                 BEGIN
; 1750          1866  4
; 1751          1867  4                 ! Exact match to runframe reg
; 1752          1868  4
; 1753          1869  4                 .reg_addr = .runframe_address;
; 1754          1870  4                 RETURN sts$k_success;
; 1755          1871  3                 END;
; 1756          1872  2             END;
; 1757          1873  2
; 1758          1874  2         ! No match
; 1759          1875  2
; 1760          1876  2         RETURN sts$k_severe;
; 1761          1877  2
; 1762          1878  1         END;                    ! End of dbg$exact_map_to_reg



                                                    0000 00000          .ENTRY  DBG$EXACT_MAP_TO_REG, Save nothing                 ; 1807
                               5E               04   C2 00002          SUBL2   #4, SP                                              ;
                                                5E   DD 00005          PUSHL   SP                                                  ; 1856
                                          04    AC   DD 00007          PUSHL   INPUT_ADDR                                          ;
                          BA   AF               02   FB 0000A          CALLS   #2, DBG$MAP_TO_REG_ADDR                             ;
                               21               50   E9 0000E          BLBC    R0, 1$                                              ;
                               50  00000000G    00   9E 00011          MOVAB   DBG$RUNFRAME+4, R0                                  ; 1863
                    50         6E               50   C3 00018          SUBL3   R0, RUNFRAME_ADDRESS, R0                            ;
       7E           00         50               01   7A 0001C          EMUL    #1, R0, #0, -(SP)                                   ;
       50           50         8E               04   7B 00021          EDIV    #4, (SP)+, R0, R0                                   ;
                                                50   D5 00026          TSTL    R0                                                  ;
                                                08   12 00028          BNEQ    1$                                                  ;
                          08   BC               6E   D0 0002A          MOVL    RUNFRAME_ADDRESS, @REG_ADDR                         ; 1869
                               50               01   D0 0002E          MOVL    #1, R0                                              ; 1870
                                                     04 00031          RET                                                         ;
                               50               04   D0 00032 1$:      MOVL    #4, R0                                              ; 1876
                                                     04 00035          RET                                                         ; 1878

; Routine Size:  54 bytes,    Routine Base:  DBG$CODE + 09C2
```

```
! MACROS:
!
!
! The keyword_table is made of four-tuple entries,
! 1) the language index (0 - n),
! 2) the number of characters in the minimal abbreviation,
! 3) the number of characters in the language name,
! 4) the language name as an ASCII string.
!
! Macro KEY_NAME formats table entries for the language name table.
! Each entry has three formals:
! 1) the ASCII string representing a language name,
! 2) the length of that ASCII string abbreviated,
! 3) the  language index for that language
!
MACRO
    KEY_NAME (KNAME, KABBREV, KEQUIV) =
                        KEQUIV, KABBREV, %CHARCOUNT (KNAME), %ASCII KNAME%;

BIND
    LANGUAGE_TABLE = UPLIT BYTE(

        KEY_NAME ('MACRO',      2, DBG$K_MACRO),
        KEY_NAME ('FORTRAN',    2, DBG$K_FORTRAN),
        KEY_NAME ('BLISS',      2, DBG$K_BLISS),
        KEY_NAME ('COBOL',      2, DBG$K_COBOL),
        KEY_NAME ('BASIC',      2, DBG$K_BASIC),
        KEY_NAME ('PLI',        2, DBG$K_PLI),
        KEY_NAME ('PASCAL',     2, DBG$K_PASCAL),
        KEY_NAME ('C',          1, DBG$K_C),
        KEY_NAME ('RPG',        2, DBG$K_RPG),
        KEY_NAME ('ADA',        2, DBG$K_ADA),
        KEY_NAME ('UNKNOWN',    3, DBG$K_UNKNOWN),

        0       ) : VECTOR [, BYTE];
```

```
1 GLOBAL ROUTINE DBG$SET_LANG (LANG_STR_DESC, LANGUAGE_MODULE) =
1 !
1 ! FUNCTION
1 !       This routine loads the pointers to the current parsing tables
1 !       with those of the new language.
1 !
1 ! INPUTS
1 !       LANG_STR_DESC   - no longer used (always 0). Eventually, this
1 !                         parameter should be eliminated (requires
1 !                         changing the 4 places this routine is called.)
1 !       LANGUAGE_MODULE - If "lang_str_desc" is zero this parameter holds the
1 !                         language number as defined in DBGLIB.REQ.
1 !
1 ! OUTPUTS
1 !       The language index of the language to which DEBUG will be set is
1 !                         returned as the routine value.
1 !
1
2   BEGIN
2   LOCAL
2       DEF_RADIX;
2
2   ! Change language setting
2   !
2   DBG$GB_LANGUAGE = .LANGUAGE_MODULE;
2   DBG$PARSER_SET_LANGUAGE (.DBG$GB_LANGUAGE);
2   DBG$NCHANGE_TO_NEW ();
2   DBG$SET_MOD_DEF ();
2   DBG$SET_STP_DEF ();
2   DBG$SET_SEARCH_DEF ();
2
2   ! Set up the default radix settings for this language.
2   !
2   def_radix = dbg$nget_trans_radix(dbg$k_default);
2   dbg$gb_radix[dbg$b_radix_input] = .def_radix;
2   dbg$gb_radix[dbg$b_radix_output] = .def_radix;
2   dbg$gb_radix[dbg$b_radix_output_over] = dbg$k_default;
2
2   RETURN .LANGUAGE_MODULE;
1   END;


                                                        .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                          05  02  00  00190 P.AAZ:  .BYTE   0, 2, 5
                  4F  52  43  41  4D  00193         .ASCII  \MACRO\
                          07  02  01  00198         .BYTE   1, 2, 7
          4E  41  52  54  52  4F  46  0019B         .ASCII  \FORTRAN\
                          05  02  02  001A2         .BYTE   2, 2, 5
                  53  53  49  4C  42  001A5         .ASCII  \BLISS\
                          05  02  03  001AA         .BYTE   3, 2, 5
                  4C  4F  42  4F  43  001AD         .ASCII  \COBOL\
                          05  02  04  001B2         .BYTE   4, 2, 5
                  43  49  53  41  42  001B5         .ASCII  \BASIC\
                          03  02  05  001BA         .BYTE   5, 2, 3
                          49  4C  50  001BD         .ASCII  \PLI\
                          06  02  06  001C0          .BYTE   6, 2, 6                                      :
              4C  41  43  53  41  50  001C3          .ASCII  \PASCAL\                                     :
                          01  01  07  001C9          .BYTE   7, 1, 1                                      :
                                  43  001CC          .ASCII  \C\                                          :
                          03  02  08  001CD          .BYTE   8, 2, 3                                      :
                          47  50  52  001D0          .ASCII  \RPG\                                        :
                          03  02  09  001D3          .BYTE   9, 2, 3                                      :
                          41  44  41  001D6          .ASCII  \ADA\                                        :
                          07  03  0A  001D9          .BYTE   10, 3, 7                                     :
          4E  57  4F  4E  4B  4E  55  001DC          .ASCII  \UNKNOWN\                                    :
                                  00  001E3          .BYTE   0                                            :

                                       LANGUAGE_TABLE=       P.AAZ


                                                     .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                000C  00000          .ENTRY  DBG$SET_LANG, Save R2,R3                     : 1914
              53  00000000G   00  9E  00002          MOVAB   DBG$GB_LANGUAGE, R3                          :
              52  00000000G   00  9E  00009          MOVAB   DBG$GB_RADIX, R2                             :
              63          08  AC  90  00010          MOVB    LANGUAGE_MODULE, DBG$GB_LANGUAGE             : 1938
              7E              63  9A  00014          MOVZBL  DBG$GB_LANGUAGE, -(SP)                       : 1939
    00000000G  00             01  FB  00017          CALLS   #1, DBG$PARSER_SET_LANGUAGE                  :
    00000000G  00             00  FB  0001E          CALLS   #0, DBG$NCHANGE_TO_NEW                       : 1940
    00000000G  00             00  FB  00025          CALLS   #0, DBG$SET_MOD_DEF                          : 1941
    00000000G  00             00  FB  0002C          CALLS   #0, DBG$SET_STP_DEF                          : 1942
    00000000G  00             00  FB  00033          CALLS   #0, DBG$SET_SEARCH_DEF                       : 1943
                              01  DD  0003A          PUSHL   #1                                           : 1947
    00000000G  00             01  FB  0003C          CALLS   #1, DBG$NGET_TRANS_RADIX                     :
               62             50  90  00043          MOVB    DEF_RADIX, DBG$GB_RADIX                      : 1948
          01   A2             50  90  00046          MOVB    DEF_RADIX, DBG$GB_RADIX+1                    : 1949
          02   A2             01  90  0004A          MOVB    #1, DBG$GB_RADIX+2                           : 1950
               50         08  AC  D0  0004E          MOVL    LANGUAGE_MODULE, R0                          : 1952
                                  04  00052          RET                                                  : 1953

; Routine Size:  83 bytes,    Routine Base:  DBG$CODE + 09F8
```

```
GLOBAL ROUTINE DBG$LANGUAGE (LANG_ENCODING) =
!
! FUNCTION
!       Return a pointer to a counted string which is
!       the name of the given language.
!       This function exists simply to consolidate this nameing
!       translation into one place.
!
! INPUTS
!       LANG_ENCODING   - The numeric encoding used internally to
!                         represent the language.  This is the same
!                         value that comes in the DST MODULE records for
!                         each language, and it is the same value that we
!                         store in DBG$GL_LANGUAGE.
!
! OUTPUTS
!       A pointer to a counted string which names the indicated language
!                         is returned as the routine value.
!

    BEGIN


    ! Just return the desired pointer.
    !
    CASE .LANG_ENCODING FROM DBG$K_MACRO TO DBG$K_UNKNOWN OF
        SET

        [DBG$K_MACRO]:
            RETURN UPLIT BYTE(%ASCIC 'MACRO');

        [DBG$K_FORTRAN]:
            RETURN UPLIT BYTE(%ASCIC 'FORTRAN');

        [DBG$K_BLISS]:
            RETURN UPLIT BYTE(%ASCIC 'BLISS');

        [DBG$K_COBOL]:
            RETURN UPLIT BYTE(%ASCIC 'COBOL');

        [DBG$K_BASIC]:
            RETURN UPLIT BYTE(%ASCIC 'BASIC');

        [DBG$K_PLI]:
            RETURN UPLIT BYTE(%ASCIC 'PLI');

        [DBG$K_PASCAL]:
            RETURN UPLIT BYTE(%ASCIC 'PASCAL');

        [DBG$K_C]:
            RETURN UPLIT BYTE(%ASCIC 'C');

        [DBG$K_RPG]:
            RETURN UPLIT BYTE(%ASCIC 'RPG');

        [DBG$K_ADA]:
            RETURN UPLIT BYTE(%ASCIC 'ADA');
```

```
; 1898          2011  2
; 1899          2012  2            [INRANGE, OUTRANGE]:
; 1900          2013  2                RETURN UPLIT BYTE(%ASCIC 'UNKNOWN');
; 1901          2014  2
; 1902          2015  2            TES;
; 1903          2016  2
; 1904          2017  1        END;


                                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                              4F  52  43  41  4D  05  001E4 P.ABA:  .ASCII  <5>\MACRO\                          ;
                      4E  41  52  54  52  4F  46  07  001EA P.ABB:  .ASCII  <7>\FORTRAN\                        ;
                              53  53  49  4C  42  05  001F2 P.ABC:  .ASCII  <5>\BLISS\                          ;
                              4C  4F  42  4F  43  05  001F8 P.ABD:  .ASCII  <5>\COBOL\                          ;
                              43  49  53  41  42  05  001FE P.ABE:  .ASCII  <5>\BASIC\                          ;
                                      49  4C  50  03  00204 P.ABF:  .ASCII  <3>\PLI\                            ;
                          4C  41  43  53  41  50  06  00208 P.ABG:  .ASCII  <6>\PASCAL\                         ;
                                              43  01  0020F P.ABH:  .ASCII  <1>\C\                              ;
                                      47  50  52  03  00211 P.ABI:  .ASCII  <3>\RPG\                            ;
                                      41  44  41  03  00215 P.ABJ:  .ASCII  <3>\ADA\                            ;
                      4E  57  4F  4E  4B  4E  55  07  00219 P.ABK:  .ASCII  <7>\UNKNOWN\                        ;


                                                                .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                               0004  00000          .ENTRY  DBG$LANGUAGE, Save R2              ; 1954
                              52  00000000'   EF  9E  00002          MOVAB   P.ABK, R2
                  0A          00         04   AC  CF  00009          CASEL   LANG_ENCODING, #0, #10             ; 2010
    0029          0024        001F            001A    0000E 1$:      .WORD   3$-1$,-                            ;
    003D          0038        0033            002E    00016                  4$-1$,-                            ;
                  0016        0047            0042    0001E                  5$-1$,-                            ;
                                                                             6$-1$,-                            ;
                                                                             7$-1$,-                            ;
                                                                             8$-1$,-                            ;
                                                                             9$-1$,-                            ;
                                                                             10$-1$,-                           ;
                                                                             11$-1$,-                           ;
                                                                             12$-1$,-                           ;
                                                                             2$-1$                              ;
                              50              62  9E  00024 2$:      MOVAB   P.ABK, R0                          ; 2013
                                                  04  00027          RET
                              50         C8   A2  9E  00028 3$:      MOVAB   P.ABA, R0                          ; 1983
                                                  04  0002C          RET                                        ; 2010
                              50         D1   A2  9E  0002D 4$:      MOVAB   P.ABB, R0                          ; 1986
                                                  04  00031          RET                                        ; 2010
                              50         D9   A2  9E  00032 5$:      MOVAB   P.ABC, R0                          ; 1989
                                                  04  00036          RET                                        ; 2010
                              50         DF   A2  9E  00037 6$:      MOVAB   P.ABD, R0                          ; 1992
                                                  04  0003B          RET                                        ; 2010
                              50         E5   A2  9E  0003C 7$:      MOVAB   P.ABE, R0                          ; 1995
                                                  04  00040          RET                                        ; 2010
                              50         EB   A2  9E  00041 8$:      MOVAB   P.ABF, R0                          ; 1998
                                                  04  00045          RET                                        ; 2010
                              50         EF   A2  9E  00046 9$:      MOVAB   P.ABG, R0                          ; 2001
```

```
                                               04 0004A          RET                                                   ; 2010
                          50      F6      A2   9E 0004B 10$:     MOVAB   P.ABH, R0                                     ; 2004
                                               04 0004F          RET                                                   ; 2010
                          50      F8      A2   9E 00050 11$:     MOVAB   P.ABI, R0                                     ; 2007
                                               04 00054          RET                                                   ; 2010
                          50      FC      A2   9E 00055 12$:     MOVAB   P.ABJ, R0                                     ;
                                               04 00059          RET                                                   ; 2017

; Routine Size:  90 bytes,    Routine Base:  DBG$CODE + 0A4B

; 1905          2018  1
; 1906          2019  1 BIND
; 1907          2020  1         deficf_name     = UPLIT BYTE(%ASCII 'DEBUG.COM'),
; 1908          2021  1         deficf_size     = %CHARCOUNT(%ASCII 'DEBUG.COM');
; 1909          2022  1 MACRO
; 1910        M 2023  1         icf_message (prefix) =
; 1911        M 2024  1
; 1912        M 2025  1         BEGIN
; 1913        M 2026  1         BIND
; 1914        M 2027  1                 enter_phrase = UPLIT BYTE(8, %ASCII 'entering'),
; 1915        M 2028  1                 exit_phrase  = UPLIT BYTE(7, %ASCII 'exiting');
; 1916        M 2029  1
; 1917        M 2030  1         LOCAL
; 1918        M 2031  1                 phrase;
; 1919        M 2032  1
; 1920        M 2033  1         IF prefix EQL 1
; 1921        M 2034  1         THEN
; 1922        M 2035  1                 phrase = enter_phrase
; 1923        M 2036  1         ELSE
; 1924        M 2037  1                 phrase = exit_phrase;
; 1925        M 2038  1
; 1926        M 2039  1
; 1927        M 2040  1         SIGNAL (dbg$_verifyicf, 3, .phrase, .fab_ptr[fab$b_fns], .fab_ptr[fab$l_fna]);  ! Info message
; 1928        M 2041  1
; 1929          2042  1         END % ;
```

```
GLOBAL ROUTINE DBG$CIS_CONNECTICF (SIGNAL_FLAG) : NOVALUE =

!++
! FUNCTIONAL DESCRIPTION:
!
!
! FORMAL PARAMETERS:
!       SIGNAL_FLAG     - TRUE if called from normal command procesing and
!                         we should signal warning message on failure.
!                         FALSE if called from setting up DEBUG initialization
!                         file. In this case just signal informational.
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
!
!       An unsigned integer longword completion code
!
! COMPLETION CODES:
!
!       NONE
!
! SIDE EFFECTS:
!
!       NONE
!
!--
    BEGIN

        LOCAL
                dummy_mess_vect,
                status,                         ! Return status
                fab_ptr : REF $FAB_DECL,        ! ptr to allocated FAB storage
                rab_ptr : REF $RAB_DECL,        ! ptr to allocated RAB storage
                ind_com_filesp : REF VECTOR [,BYTE];    ! Filespec counted string

        ind_com_filesp = .dbg$gl_ind_com_file;

        ! Allocate FAB and RAB storage

        fab_ptr = dbg$get_memory ((fab$c_bln + 3)/ %UPVAL);
        rab_ptr = dbg$get_memory ((rab$c_bln + 3)/ %UPVAL);

        ! Initialize the FAB and the RAB

        $FAB_INIT (FAB=.fab_ptr, FAC=GET, FNA=.ind_com_filesp + 1, FNS=.ind_com_filesp[0],
                   DNA=deficf_name, DNS=deficf_size);
        $RAB_INIT (RAB=.rab_ptr, FAB=.fab_ptr);

        ! Put them on the command input stream

```

```
dbg$cis_add (.rab_ptr, 0, cis_rab, 0, 0);

! Set up the local define list for the command procedure.
!
IF NOT dbg$def_pr_entry (dummy_mess_vect)
THEN

    ! Signal the error.
    !
    BEGIN
    EXTERNAL ROUTINE
        lib$signal: ADDRESSING_MODE(GENERAL);
    BUILTIN
        CALLG;
    CALLG (.dummy_mess_vect, lib$signal);
    END;

! Open and connect the file
!
status = $OPEN (FAB=.fab_ptr);
IF NOT .status
THEN
        BEGIN


        LOCAL
                msg_desc : BLOCK [8,BYTE];

        msg_desc[dsc$w_length]  = .fab_ptr[fab$b_fns];
        msg_desc[dsc$a_pointer] = .fab_ptr[fab$l_fna];


        ! Flag link for removal so we won't try to read from it again

        dbg$gl_cishead[cis$v_rem_flag] = 1;

        IF .signal_flag
        THEN
            SIGNAL (shr$_openin + dbg_fac_code, 1, msg_desc,
                .fab_ptr[fab$l_sts], .fab_ptr[fab$l_stv])
        ELSE
            BEGIN
            SIGNAL (dbg$_unaopnini, 1, msg_desc,
                .fab_ptr[fab$l_sts], .fab_ptr[fab$l_stv]);
            RETURN;
            END;
        END;

! Connect the RAB to the just opened FAB

status = $CONNECT (RAB=.rab_ptr);
IF NOT .status
THEN
        BEGIN
        LOCAL
                msg_desc : BLOCK [8,BYTE];
```

```
; 2045        2157 3                    msg_desc[dsc$w_length] = .fab_ptr[fab$b_fns];
; 2046        2158 3                    msg_desc[dsc$a_pointer] = .fab_ptr[fab$l_fna];
; 2047        2159 3
; 2048        2160 3
; 2049        2161 3                    ! Flag link for removal so we won't try to read from it again
; 2050        2162 3
; 2051        2163 3                    dbg$gl_cishead[cis$v_rem_flag] = 1;
; 2052        2164 3
; 2053        2165 3                    IF .signal_flag
; 2054        2166 3                    THEN
; 2055        2167 3                        SIGNAL (shr$_openin + dbg_fac_code, 1, msg_desc,
; 2056        2168 3                            .fab_ptr[fab$l_sts], .fab_ptr[fab$l_stv])
; 2057        2169 3                    ELSE
; 2058        2170 3                        SIGNAL (dbg$_unaopnini, 1, msg_desc,
; 2059        2171 3                            .fab_ptr[fab$l_sts], .fab_ptr[fab$l_stv]);
; 2060        2172 3
; 2061        2173 3
; 2062        2174 2                    END;
; 2063        2175 2
; 2064        2176 2            IF .dbg$gb_def_out [out_verify]
; 2065        2177 2            THEN
; 2066        2178 2                    icf_message(1);
; 2067        2179 2
; 2068        2180 2            RETURN;
; 2069        2181 2
; 2070        2182 1 END;               ! End of dbg$cis_connecticf


                                                              .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

        4D  4F  43  2E  47  55  42  45  44  00221 P.ABL:     .ASCII  \DEBUG.COM\                                   ;
                                        08  0022A P.ABM:     .BYTE   8                                             ;
            67  6E  69  72  65  74  6E  65  0022B            .ASCII  \entering\                                    ;
                                        07  00233 P.ABN:     .BYTE   7                                             ;
                67  6E  69  74  69  78  65  00234            .ASCII  \exiting\                                     ;

                                                  DEFICF_NAME=        P.ABL
                                                  DEFICF_SIZE=        9
                                                  ENTER_PHRASE=       P.ABM
                                                  EXIT_PHRASE=        P.ABN
                                                             .EXTRN  LIB$SIGNAL

                                                             .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                   0FFC 00000                .ENTRY  DBG$CIS_CONNECTICF, Save R2,R3,R4,R5,R6,R7,-; 2043
                                                                     R8,R9,R10,R11                                 ;
                    5B 00000000G   00  9E 00002              MOVAB   DBG$GL_CISHEAD, R11                           ;
                    5A 00000000G   00  9E 00009              MOVAB   DBG$GET_MEMORY, R10                           ;
                    59 00000000G   00  9E 00010              MOVAB   LIB$SIGNAL, R9                                ;
                    5E             0C  C2 00017              SUBL2   #12, SP                                       ;
                    58 00000000G   00  D0 0001A              MOVL    DBG$GL_IND_COM_FILE, IND_COM_FILESP           ; 2085
                                   14  DD 00021              PUSHL   #20                                           ; 2089
                    6A             01  FB 00023              CALLS   #1, DBG$GET_MEMORY                            ;
                    56             50  D0 00026              MOVL    R0, FAB_PTR                                   ;
                                   11  DD 00029              PUSHL   #17                                           ; 2090
                    6A             01  FB 0002B              CALLS   #1, DBG$GET_MEMORY                            ;
```

```
                                       57           50  D0 0002E           MOVL    R0, RAB_PTR                         ;
  0050   8F           00               6E           00  2C 00031           MOVC5   #0, (SP), #0, #80, (FAB_PTR)        ; 2095
                                                    66     00038                                                       ;
                                       66     5003  8F  B0 00039           MOVW    #20483, (FAB_PTR)                   ;
                                  16   A6           02  90 0003E           MOVB    #2, 22(FAB_PTR)                     ;
                                  1F   A6           02  90 00042           MOVB    #2, 31(FAB_PTR)                     ;
                                  2C   A6       01  A8  9E 00046           MOVAB   1(R8), 44(FAB_PTR)                  ;
                                  30   A6 00000000' EF  9E 0004B           MOVAB   DEFICF_NAME, 48(FAB_PTR)            ;
                                  34   A6           68  90 00053           MOVB    (IND_COM_FILESP), 52(FAB_PTR)       ;
                                  35   A6           09  90 00057           MOVB    #9, 53(FAB_PTR)                     ;
  0044   8F           00               6E           00  2C 0005B           MOVC5   #0, (SP), #0, #68, (RAB_PTR)        ; 2096
                                                    67     00062                                                       ;
                                       67     4401  8F  B0 00063           MOVW    #17409, (RAB_PTR)                   ;
                                  3C   A7           56  D0 00068           MOVL    FAB_PTR, 60(RAB_PTR)                ;
                                                    7E  7C 0006C           CLRQ    -(SP)                               ; 2100
                                                    01  DD 0006E           PUSHL   #1                                  ;
                                                    7E  D4 00070           CLRL    -(SP)                               ;
                                                    57  DD 00072           PUSHL   RAB_PTR                             ;
                               0000V   CF           05  FB 00074           CALLS   #5, DBG$CIS_ADD                     ;
                                                    5E  DD 00079           PUSHL   SP                                  ; 2104
                           00000000G   00           01  FB 0007B           CALLS   #1, DBG$DEF_PR_ENTRY                ;
                                       04           50  E8 00082           BLBS    R0, 1$                              ;
                                       69       00  BE  FA 00085           CALLG   @DUMMY_MESS_VECT, LIB$SIGNAL        ; 2114
                                                    56  DD 00089 1$:       PUSHL   FAB_PTR                             ; 2119
                           00000000G   00           01  FB 0008B           CALLS   #1, SYS$OPEN                        ;
                                       52           50  D0 00092           MOVL    R0, STATUS                          ;
                                       3A           52  E8 00095           BLBS    STATUS, 3$                          ; 2120
                                  04   AE       34  A6  9B 00098           MOVZBW  52(FAB_PTR), MSG_DESC               ; 2128
                                  08   AE       2C  A6  D0 0009D           MOVL    44(FAB_PTR), MSG_DESC+4             ; 2129
                                       50           6B  D0 000A2           MOVL    DBG$GL_CISHEAD, R0                  ; 2134
                                  12   A0           01  88 000A5           BISB2   #1, 18(R0)                          ;
                                       14       04  AC  E9 000A9           BLBC    SIGNAL_FLAG, 2$                     ; 2136
                                       7E       08  A6  7D 000AD           MOVQ    8(FAB_PTR), -(SP)                   ; 2139
                                                0C  AE  9F 000B1           PUSHAB  MSG_DESC                            ; 2138
                                                    01  DD 000B4           PUSHL   #1                                  ;
                                          00021098  8F  DD 000B6           PUSHL   #135320                             ;
                                       69           05  FB 000BC           CALLS   #5, LIB$SIGNAL                      ;
                                                    11  11 000BF           BRB     3$                                  ;
                                       7E       08  A6  7D 000C1 2$:       MOVQ    8(FAB_PTR), -(SP)                   ; 2143
                                                0C  AE  9F 000C5           PUSHAB  MSG_DESC                            ; 2142
                                                    01  DD 000C8           PUSHL   #1                                  ;
                                          00028683  8F  DD 000CA           PUSHL   #165507                             ;
                                                    66  11 000D0           BRB     7$                                  ;
                                                    57  DD 000D2 3$:       PUSHL   RAB_PTR                             ; 2150
                           00000000G   00           01  FB 000D4           CALLS   #1, SYS$CONNECT                     ;
                                       52           50  D0 000DB           MOVL    R0, STATUS                          ;
                                       38           52  E8 000DE           BLBS    STATUS, 6$                          ; 2151
                                  04   AE       34  A6  9B 000E1           MOVZBW  52(FAB_PTR), MSG_DESC               ; 2157
                                  08   AE       2C  A6  D0 000E6           MOVL    44(FAB_PTR), MSG_DESC+4             ; 2158
                                       50           6B  D0 000EB           MOVL    DBG$GL_CISHEAD, R0                  ; 2163
                                  12   A0           01  88 000EE           BISB2   #1, 18(R0)                          ;
                                       11       04  AC  E9 000F2           BLBC    SIGNAL_FLAG, 4$                     ; 2165
                                       7E       08  A6  7D 000F6           MOVQ    8(FAB_PTR), -(SP)                   ; 2168
                                                0C  AE  9F 000FA           PUSHAB  MSG_DESC                            ; 2167
                                                    01  DD 000FD           PUSHL   #1                                  ;
                                          00021098  8F  DD 000FF           PUSHL   #135320                             ;
                                                    0F  11 00105           BRB     5$                                  ;
```

```
7E          08  A6  7D 00107 4$:     MOVQ    8(FAB_PTR), -(SP)                  ; 2171
            0C  AE  9F 0010B         PUSHAB  MSG_DESC                           ; 2170
                01  DD 0010E         PUSHL   #1                                 ;
      00028683  8F  DD 00110         PUSHL   #165507                            ;
69              05  FB 00116 5$:     CALLS   #5, LIB$SIGNAL                     ;
1B 00000000G    00  E9 00119 6$:     BLBC    DBG$GB_DEF_OUT+2, 8$               ; 2176
50 00000000'    EF  9E 00120         MOVAB   ENTER_PHRASE, PHRASE               ; 2178
            2C  A6  DD 00127         PUSHL   44(FAB_PTR)                        ;
7E          34  A6  9A 0012A         MOVZBL  52(FAB_PTR), -(SP)                 ;
                50  DD 0012E         PUSHL   PHRASE                             ;
                03  DD 00130         PUSHL   #3                                 ;
      0002808B  8F  DD 00132         PUSHL   #163979                            ;
69              05  FB 00138 7$:     CALLS   #5, LIB$SIGNAL                     ;
                    04 0013B 8$:     RET                                        ; 2182

; Routine Size:  316 bytes,    Routine Base:  DBG$CODE + 0AA5
; 2072       2183   1 GLOBAL ROUTINE dbg$cis_remove (exit_flag) : NOVALUE =
; 2073       2184   1 !++
; 2074       2185   1 ! FUNCTIONAL DECSRIPTION:
; 2075       2186   1 !       Removes the top link from the command input stream and delete the
; 2076       2187   1 !       storage for it. If the link has additional dynamic storage related to
; 2077       2188   1 !       it, such as a FAB,RAB, input buffer etc., that storage is freed also.
; 2078       2189   1 !       Note - this routine now just calls the routine DBG$NCIS_REMOVE in
; 2079       2190   1 !       the module DBGNEXCTE.
; 2080       2191   1 !
; 2081       2192   1 ! FORMAL PARAMETERS:
; 2082       2193   1 !
; 2083       2194   1 !       exit_flag -     TRUE if called from EXIT command.
; 2084       2195   1 !
; 2085       2196   1 ! IMPLICIT INPUTS:
; 2086       2197   1 !       The head of the command input stream
; 2087       2198   1 !
; 2088       2199   1 ! IMPLICIT OUTPUTS:
; 2089       2200   1 !       None
; 2090       2201   1 !
; 2091       2202   1 ! ROUTINE VALUE:
; 2092       2203   1 !       None
; 2093       2204   1 !
; 2094       2205   1 ! SIDE EFFECTS:
; 2095       2206   1 !       The head of the command input stream is reset to what was the
; 2096       2207   1 !       "next" link before this routine was called. If SET OUTPUT VERIFY,
; 2097       2208   1 !       then a message is generated saying we are exiting the indirect
; 2098       2209   1 !       command file.
; 2099       2210   1 !--
; 2100       2211   1
; 2101       2212   2     BEGIN
; 2102       2213   2
; 2103       2214   2     LOCAL
; 2104       2215   2         message_vect;           ! Dummy message argument vector.
; 2105       2216   2
; 2106       2217   2     ! Call the 'new debugger' routine. This returns a condition code
; 2107       2218   2     ! of 'severe', together with an error message vector, if something
; 2108       2219   2     ! goes wrong.
; 2109       2220   2     !
; 2110       2221   2     IF NOT dbg$ncis_remove (.exit_flag, message_vect)
; 2111       2222   2     THEN
; 2112       2223   3         BEGIN
; 2113       2224   3         ! Set up to signal error.
; 2114       2225   3         !
; 2115       2226   3         EXTERNAL ROUTINE
; 2116       2227   3             lib$signal : ADDRESSING_MODE (GENERAL);
; 2117       2228   3         BUILTIN
; 2118       2229   3             callg;
; 2119       2230   3         callg (.message_vect, lib$signal);
; 2120       2231   2         END;
; 2121       2232   1     END;


                                        0000 00000          .ENTRY  DBG$CIS_REMOVE, Save nothing          ; 2183
                   5E              04   C2 00002            SUBL2   #4, SP                                 ;
```

```
                                   5E  DD 00005          PUSHL   SP                                   ; 2221
                            04     AC  DD 00007          PUSHL   EXIT_FLAG                            ;
             00000000G  00         02  FB 0000A          CALLS   #2, DBG$NCIS_REMOVE                  ;
                        08         50  E8 00011          BLBS    R0, 1$                               ;
             00000000G  00  00     BE  FA 00014          CALLG   @MESSAGE_VECT, LIB$SIGNAL            ; 2230
                                       04 0001C 1$:      RET                                          ; 2232

; Routine Size:  29 bytes,    Routine Base:  DBG$CODE + 0BE1
```

```
GLOBAL ROUTINE dbg$cis_add (pointer, length, type,
                            repeat_count, while_clause): NOVALUE =
!++
! FUNCTIONAL DESCRIPTION:
!       Adds a link to the command input stream
!       Note - this routine now just calls the routine DBG$NCIS_ADD in
!       the module DBGNEXCTE.
!
! FORMAL PARAMETERS:
!       pointer -       The address of either a buffer or a RAB to be placed
!                       in the dsc$a_pointer field of the new link.
!       length  -       The length of the above buffer. (0 for RAB)
!       type    -       The type of the link to be added
!       repeat_count -  For a link of type "doloop" [Created during processing
!                       of REPEAT N TIMES ( ... ) command], this represents the
!                       number of remaining iterations.
!       while_clause -  For a link of type "while", this points to a counted
!                       ascii string with the while clause.
!
! IMPLICIT INPUTS:
!       The head of the command input stream
!
! IMPLICIT OUTPUTS:
!       None
!
! ROUTINE VALUE:
!       None
!
! SIDE EFFECTS:
!       None
!--

    BEGIN
    LOCAL
        message_vect;   ! Holds message argument vector.

    ! DBG$NCIS_ADD will return 'success' (1) if all goes well.
    !
    IF NOT dbg$ncis_add (.pointer, .length, .type,
                         .repeat_count, .while_clause, 0,
                          message_vect)
    THEN
        BEGIN
        ! Set up to signal error.
        !
        EXTERNAL ROUTINE
            lib$signal : ADDRESSING_MODE (GENERAL);
        BUILTIN
            callg;
        callg (.message_vect, lib$signal);
        END;
    END;
```

```
                                   0000 00000         .ENTRY  DBG$CIS_ADD, Save nothing          ; 2233
                5E              04   C2 00002         SUBL2   #4, SP                             ;
                                5E   DD 00005         PUSHL   SP                                 ; 2271
                                7E   D4 00007         CLRL    -(SP)                              ;
                7E        10    AC   7D 00009         MOVQ    REPEAT_COUNT, -(SP)                ; 2272
                7E        08    AC   7D 0000D         MOVQ    LENGTH, -(SP)                      ; 2271
                          04    AC   DD 00011         PUSHL   POINTER                            ;
      00000000G 00              07   FB 00014         CALLS   #7, DBG$NCIS_ADD                   ;
                08              50   E8 0001B         BLBS    R0, 1$                             ;
      00000000G 00        00    BE   FA 0001E         CALLG   @MESSAGE_VECT, LIB$SIGNAL          ; 2282
                                  04 00026 1$:        RET                                        ; 2284

; Routine Size:  39 bytes,    Routine Base:  DBG$CODE + 0BFE
; 2176          2285  1 MACRO
; 2177        M 2286  1         IF_SIGNAL (code)  =
; 2178        M 2287  1             IF .signal_flag NEQ 0
; 2179        M 2288  1             THEN
; 2180        M 2289  1                 BEGIN
; 2181        M 2290  1                     IF NOT
; 2182        M 2291  1                         ( IF %LENGTH GTR 1
; 2183        M 2292  1                           THEN dbg$nout_info (code, %REMAINING)
; 2184        M 2293  1                           ELSE dbg$nout_info (code))
; 2185        M 2294  1                     THEN
; 2186        M 2295  1                         BEGIN
; 2187        M 2296  1                             .signal_flag = (IF %LENGTH GTR 1
; 2188        M 2297  1                                             THEN
; 2189        M 2298  1                                                 dbg$nmake_arg_vect (code, %REMAINING)
; 2190        M 2299  1                                             ELSE
; 2191        M 2300  1                                                 dbg$nmake_arg_vect (code));
; 2192        M 2301  1                             RETURN sts$k_severe;
; 2193        M 2302  1                         END
; 2194        M 2303  1                 END
; 2195        M 2304  1             ELSE
; 2196        M 2305  1                 BEGIN
; 2197        M 2306  1                     IF %LENGTH GTR 1
; 2198        M 2307  1                     THEN
; 2199        M 2308  1                         SIGNAL (code, %REMAINING)
; 2200        M 2309  1                     ELSE
; 2201        M 2310  1                         SIGNAL (code)
; 2202          2311  1                 END %;
; 2203          2312  1
; 2204          2313  1 MACRO
; 2205        M 2314  1         SET_FLAG (param_num) =
; 2206        M 2315  1             LOCAL
; 2207        M 2316  1                 signal_flag;
; 2208        M 2317  1
; 2209        M 2318  1             signal_flag = (IF actualcount () GTR param_num
; 2210        M 2319  1                            THEN
; 2211        M 2320  1                                 actualparameter (actualcount())
; 2212        M 2321  1                            ELSE
; 2213          2322  1                                 0) %;
; 2214          2323  1
; 2215          2324  1 END
; 2216          2325  0 ELUDOM



                                                                        .EXTRN  LIB$SIGNAL

;                                       PSECT SUMMARY
;
;       Name                         Bytes                          Attributes
;
;   DBG$PLIT                          571  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;   DBG$CODE                         3109  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;   DBG$OWN                            32  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;                               Library Statistics
;
;                                                      -------- Symbols --------         Pages         Processing
;           File                                       Total      Loaded   Percent       Mapped        Time
;
;   _$255$DUA28:[SYSLIB]LIB.L32;1                      18619         86         0        1000            00:01.9
;   _$255$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1                 32          0         0           7            00:00.1
;   _$255$DUA28:[DEBUG.OBJ]DBGLIB.L32;1                 1545        124         8          97            00:01.9
;   _$255$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1
;                                                        418         11         2          31            00:00.3
;   _$255$DUA28:[DEBUG.OBJ]DBGMSG.L32;1                  386         36         9          22            00:00.3
;   _$255$DUA28:[DEBUG.OBJ]DBGGEN.L32;1                  150         30        20          12            00:00.3




;                                        COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:DBGLEVEL1/OBJ=OBJ$:DBGLEVEL1 MSRC$:DBGLEVEL1/UPDATE=(ENH$:DBGLEVEL1)

; Size:          3109 code + 603 data bytes
; Run Time:         01:03.9
; Elapsed Time:     03:10.8
; Lines/CPU Min:     2181
; Lexemes/CPU-Min: 18423
; Memory Used:  304 pages
; Compilation Complete
```

0084 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

DBGIFTHEN
LIS

DBGLANVEC
LIS

DBGLANGOP
LIS

DBGGEN
LIS

DBGLEVEL1
LIS

0085 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

DBGLEVEL3
LIS

DBGLIB
LIS